

Find & Replace It!

Version 2.1.2

User's Manual

Last update: September 2013

Copyright © 2009-2013 dProg – Philippe Docourt
All rights reserved.

Table of Content

1 About Find & Replace It!	4
1.1 Summary	4
1.2 Main Features	5
1.3 Supported Platforms	5
1.4 Getting Started	6
1.5 Support and Services	6
1.6 What to Do in Case Of Problems?	6
1.7 Known Issues and Limitations	7
2 Installation	8
3 Activation	9
3.1 Demo Version	9
3.2 Full Version	10
4 User Interface	12
5 Two-Minute Guide to Replacing Text in Files	14
6 Functions	15
6.1 Selecting Files to Process	15
6.2 Detecting or Selecting the Encoding of Files	16
6.3 Converting the Encoding of Files	17
6.4 Searching for an Expression in Files	18
6.5 Replacing a Given Expression in Files	19
6.6 Executing Several Replacements in One Shot	20
6.7 Editing Find & Replace Preferences	20
6.7.1 Changing the Encoding Detection Behavior	21
6.7.2 Doing Backups of Files before Modification	21
6.8 Using the Find & Replace Preview	21
6.8.1 Previewing as Plain Text	22
6.8.2 Previewing as HTML	24
6.9 Using the Regular Expression Editor	25
6.10 Advanced Replacements	26
6.10.1 Using of Captured Texts within Replaced Texts	26
6.10.2 Processing the Captured Texts with Built-In Functions	28
6.10.3 Processing the Replaced Texts by Script	28
6.11 Using the Output Window	32
6.12 Editing the Preferences	32
6.13 Using the Command Line	34
6.14 Using Find & Replace It! Documents	35
7 Tips and Tricks	36
7.1 Multi-document Tabs	36
7.2 Working with Text Areas	36
7.2.1 Navigating text	36
7.2.2 Editing Text	36
7.2.3 Undo/Redo Changes	37
7.2.4 Changing Display Properties	37
7.2.5 Searching for Text in Text Areas	37
7.3 Using Logical Folders for Searching Files	38
7.4 Using Local Storage for Settings	38
7.5 Multi File Selection in the Found Files List	39
7.6 Getting Examples	39
7.7 Debugging Script	40
8 Regular Expressions	41
8.1 Introduction	41
8.2 Characters and Abbreviations for Sets of Characters	42
8.3 Sets of Characters	43

8.4 Quantifiers.....	43
8.5 Capturing Text.....	44
8.6 Assertions.....	44
8.7 Wildcard Matching.....	45
8.8 Notes for Perl Users.....	45
8.9 Examples.....	46
9 Licensing Information.....	47
9.1 End User Licenses.....	47
9.1.1 End User License for NON App Store Customers.....	47
9.1.2 End User License for App Store Customers.....	47
9.2 Third Party Licenses and Credits.....	48
9.3 Trademarks.....	49

1 About Find & Replace It!

1.1 Summary



Find & Replace It! is a high-end solution to find files and execute search and replace operations across multiple files. It allows performing very complex batch replacements inside text files of any size. It supports regular expression syntax and dozens of encodings. It has scripting capabilities which allow transforming on the fly the replacement text for every found string. It even handles batch processing of the encoding of files, as well as of types of end-of-lines.

If you regularly work with text files, if you make websites, or develop software, *Find & Replace It!* allows you to maintain and transform the contents of your files in a few clicks. Do you have to rename a page in a website and thus need to edit URLs in hundreds of files? Simply select the root directory of your website and indicate the string to replace. Then, the replacement can be made across your entire website in just one click. Do you need to modify files headers in some parts of the source code of your application? *Find & Replace It!* allows to finely filter the files to be processed or not. It also provides tools to introduce contextual information into your replacement string, for instance: the current date, the name of the file where the replacement is in progress, etc. For each batch-replacement, it provides statistics about the number of replacements within every modified file, and counts the number of processed files. Of course, all classical features of such a tool, like the possibility to save your job, backup modified files, export the replacement report, and much more are available.

So far *Find & Replace It!* looks like a classical search & replace utility like many others. However here are five key points that make the difference with its competitors:

- Handles more than 50 different kinds of text encodings. If you have troubles with any kind of Unicode data like UTF-8 encoded files, any Chinese content, or any other non ASCII contents, *Find & Replace It!* is the solution for you. It also handles the different kinds of end-of-line.
- Handles huge files. If you want to process any kind of files containing tons of text, like a log of 10 GB, *Find & Replace It!* will do it for you.
- Provides a regular expression editor. This editor offers tools to easily build advanced regular expressions even if you are a newbie.
- Provides a preview for found and replaced text. With the preview window you can visualize in one glance what are the strings that match a given expression, what will be the replacement text, and even mix both views. It gives you an immediate feed-back on what is going to change, which makes easy to understand the impact of your replacement within a file without actually modifying its content. Such a realtime visualization is especially useful whenever you want to build a complex expression to search for.
- Makes it possible to dynamically adjust the replacement text. It provides many ways of dynamically adjusting the replacement text according to the context. For instance, it is possible to reuse a fragment of the found expression into the replacement text, do arithmetic operations on found text, insert the path or the name of the processed file, apply conditional operations on the replacement string (JavaScript interface), manipulate dates, etc.

These key features associated to many others makes it one of the most feature rich tool to find and replace regular expressions over multiple files. It's also a powerful tool for converting the text encodings (charsets conversion, including the Byte Order Marks), or the end-of-line delimiters, across multiple files. In addition, it's probably the only software in this category that is portable!

The GUI of *Find & Replace It!* consists of several panels that you can arrange in any way you want. You can dock panels side by side, arrange them in tabs, or make them float. Adjust the windows layout to the way you like to work. *Find & Replace It!* comes with four pre-arrange layouts that can be customized the way you want depending on the size of your screen. You can switch from one layout to another in a simple click.

1.2 Main Features

The list below describes some of the most important characteristics of *Find & Replace It!*:

- Find and replace strings across many files at once
- Execute several replace operations at once (i.e. ability to run a sequence of several distinct replacements in one shot)
- Supports regular expression syntax
- Allows multi-line searching
- Supports many text encodings, including Unicode (e.g. UTF-8)
- Preserves line endings while processing files
- Preserves BOM while processing Unicode files
- Allows to perform dynamic text replacements based on found expression captures
- Provides built-in processing function for dynamic replacements (e.g. convert captured expressions to lower case, Base64 encoding, Hex encoding, UTF-8 encoding, etc.)
- Provides a JavaScript like interface to customize replacements on the fly by script processing
- Displays matched expressions reports for file search/replace operations
- Full featured dynamic preview of matched expressions and replacements
- Provides tools for converting text encoding
- Provides tools for converting line endings (Windows, Unix, Macintosh, Unicode)
- Detects text encoding and line endings of files
- Provides advanced filtering options for selecting files that need to be processed, including file name filters and file path exclusion filters
- Allows to load and save expressions to find, replacement definitions and file filters
- Handles huge files (> 10 GB)
- Regular expression editor / tester
- Fully multi-threaded for maximum performance, and great responsiveness
- Allows to cancel long operations
- GUI is totally modular
- Creates backup of changed files if required
- Exports the search and replace reports
- Cross-platform: Windows, Mac OS X and Linux

1.3 Supported Platforms

Please refer to the appropriate installation instructions available at:

<http://www.dprog.ch/multimedia/findreplaceit-docs/2.1/install.html#title-supported-platforms>

The supported platforms varies over time. Therefore you are kindly requested to use the appropriate link, with the appropriate version number (findreplaceit-docs/<version>/install.html), to get the correct notice.

1.4 Getting Started

The installation instructions are available [here](#).

Find & Replace It! is protected by a licenses system. Hence to get a license for the full version of the product you should activate your copy of the software with a serial code called *Activation Key*. Without the activation, the software can still be run in demo mode with some limitations. You'll find more information in the [Activation chapter](#).

In order to quickly get started with the main components of the GUI, [here is a 2-minutes guide](#).

All references to the online documentation and resources are listed on this [documentation page](#).

1.5 Support and Services

For general information, please visit our website at: <http://www.dprog.ch>

The chapter [End User License](#) describes the licensing terms for *Find & Replace It!*. If you have any questions about pricing and/or license terms, don't hesitate to write to us at: order@dprog.ch

All support requests regarding software usage as well as general questions about demo version must be addressed to: support@dprog.ch

Please note that support might be only available to registered customers or users who have a valid license for their software copies. Moreover we kindly request our customers to use the online form for posting support requests. This form is accessible through:

<http://www.dprog.ch/home/support/>

Any User who has a valid license is eligible for free support services.

Finally, you might read the [online Terms of Use statement](#) for details about services provided by dProg – Philippe Docourt.

1.6 What to Do in Case Of Problems?

In case you encounter a reproducible crash, we might request you to [trace the application's activity](#). This can be done by running *Find & Replace It!* from the command line with the `-trace` argument. This produces a trace file that might help us to identify when the crash occurs.

In order to trace the application activity you should go through the following steps:

1. Open a terminal
2. Go to the software installation directory: `cd <path/to/FindReplaceIt/binary>` (on Mac OS X: `cd '/Applications/Find & Replace It!.app/Contents/MacOS'`)
3. Type: `./FindReplaceIt -trace [<path/to/trace/file>.xml]`

The last command enables an advanced logging for debug purpose in case of problems. Without the optional argument that specify a custom path for the trace file, this command will generate by default a trace file named `trace.xml` alongside the application executable file (on Mac OS X: `'/Applications/Find & Replace It!.app/Contents/MacOS/trace.xml'`). This file contains absolutely no personal information about you, but data about internal running of the software. We kindly request you to send us this trace file called `trace.xml` at support@dprog.ch, preferably in a compressed format.

Important Note: If you do not have write permission into your installation directory, use the additional parameter, which comes after the `-trace` argument, in order to redirect the output into a directory that is writable. Alternatively you might copy/paste your installation into a folder where you can have write access to be sure the trace file will be generated.

1.7 Known Issues and Limitations

The text-encoding detection uses heuristics that do not always provide accurate results except for all Unicode encodings (i.e.: UTF-8, UTF-16, UTF-32). Since version 1.0 the charset detection has been greatly improved. Since this version we use the ICU library from IBM. However, the process is partly statistical in nature, and the results can not be guaranteed to always be correct.

For performance reasons the [Find & Replace Preview](#) has a content limit of 100'000'000 characters. By default this limit is fixed to 5'000'000 to increase the preview speed. This can be changed through the [preferences panel](#).

For performance reasons it is not possible to search for an expression longer than 50'000'000 characters. By default this limit is fixed to 100'000 to increase speed of strings replacement. This can be changed through the [preferences panel](#).

The product activation requires an Internet connection. There is no way to proceed with activation by phone or any other communication channel.

For the sake of performance, the end-of-line detection reads a maximum of 10 MB of the analyzed file.

The file creation date is not preserved when a file is modified by a search and replace operation.

It is possible to use the application from the command line, however it requires a graphical environment.

The product activation requires an Internet connection. There is no other way to activate your copy of the software.

Unfortunately, *Find & Replace It!* cannot handle MS Word documents, Pages, PDF or others. It is very powerful to manipulate text files, but is limited to thereto. The binary documents like those of MS Word are usually described in a proprietary format. For example, it is not directly possible to extract the text content of a paragraph from a Word document. Indeed, in this case, the text of the paragraph does not appear in clear text in the contents of the DOC file. To get the plain text one must first process the document. Moreover, these manipulations are extremely complex (formatting, tables, etc.) and such a feature will far exceed the range of a simple utility software.

2 Installation

Please refer to the appropriate installation instructions available at:

<http://www.dprog.ch/multimedia/findreplaceit-docs/2.1/install.html>

These instructions might vary from a version to another. Therefore you are kindly requested to use the appropriate link, with the appropriate version number (findreplaceit-docs/<version>/install.html), to get the correct installation notice.

The installation notice covers the supported platforms, how to install the software under Windows, Mac and Linux.

3 Activation

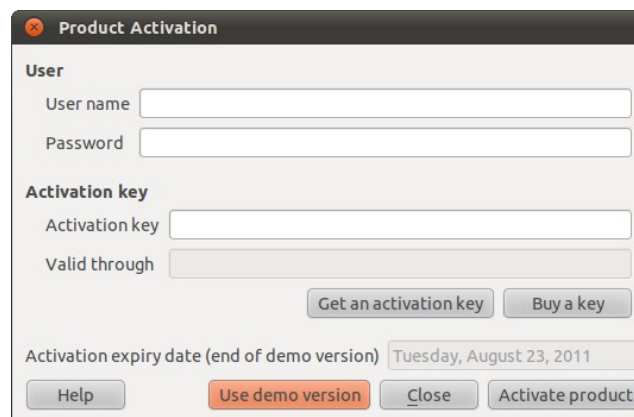
There are two ways of using *Find & Replace It!*, either in demo mode or in full-featured mode. Every time you start the application, a dialog window will ask you to activate your copy of *Find & Replace It!* or to run it in demo mode.

Note: For people who bought *Find & Replace It!* on the App Store from Apple, there is no activation key required to use the software, but there is no demo version available either. In that case this chapter does not concern you.

3.1 Demo Version

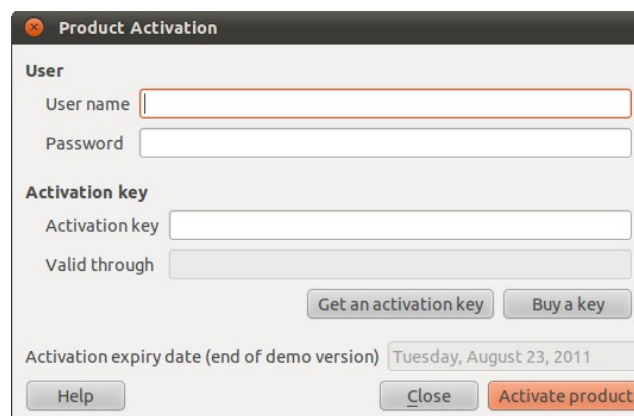
In demo mode, you don't need to proceed with activation, meaning you don't need an activation key, or a login on our website.

The demo version comes with all major features except that you can neither save your replacement configuration files, nor replace text directly in files. However, the preview window let you see the result of replacements in a read-only mode.

The screenshot shows a 'Product Activation' dialog box. It has a title bar with a close button. The main area contains several input fields: 'User name', 'Password', 'Activation key', and 'Valid through'. Below these are two buttons: 'Get an activation key' and 'Buy a key'. At the bottom, there is a text field for 'Activation expiry date (end of demo version)' showing 'Tuesday, August 23, 2011'. Below this are four buttons: 'Help', 'Use demo version' (highlighted in orange), 'Close', and 'Activate product'.

The demo version is available until August 23, 2011.

The activation window shows you an expiry date for activating the software. After this date, the software will not start any more without being activated first. Until this date, you can simply refuse the activation by clicking on **Use demo version**, and thus use the software in demo mode. Once that the demo period expired the activation button is replaced with a button named **Close**.

This screenshot shows the same 'Product Activation' dialog box, but the 'Use demo version' button has been replaced with a 'Close' button. The 'Activate product' button remains. The expiry date is still 'Tuesday, August 23, 2011'. The 'User name' field now has a red border, indicating it is required.

The demo period ended on August 23, 2011. There is no other choice than activating the product.

3.2 Full Version

The activation process registers your activation key, so that you become its owner, and nobody else can use it. An activation is necessary to gain access to the full version of the product.

The activation requires the following steps:

1. Enter your login for www.dprog.ch website (i.e. you have to register first on our website).
2. Get an activation key or recover a previous activation key:
 1. You can get a 30-day trial key for activating your copy of the software by clicking on **Get an activation key**. Note that if you already had activated the software on this machine, you'll get your previous activation key in return instead of a new trial key, unless your hardware or software configuration has changed (see #3).
 2. You can buy an activation key by clicking on **Buy a key**. You should receive your activation key by e-mail once you have bought it.
 3. You can recover the last activation key used on your computer (e.g.: after a new install) by clicking on **Get an activation key** (this works either for retrieving a trial key or a perpetual key). Then, you should get a message announcing that your activation key has been synchronized. If not, it means that your hardware or software configuration has changed since your previous activation. If the software cannot find a key related to your hardware you'll get a new 30-day trial key (see #1).

The previous activation key used was recovered by clicking the button Get an activation key (e.g. after installing an update).

3. Type or paste your activation key in the appropriate field.
4. Press **Activate product** and wait for the answer.
 1. The activation process requires a connection to Internet.
 2. In case of success, the activation window will be automatically closed. Otherwise, you might try again later. In case of problem contact [our support](#).

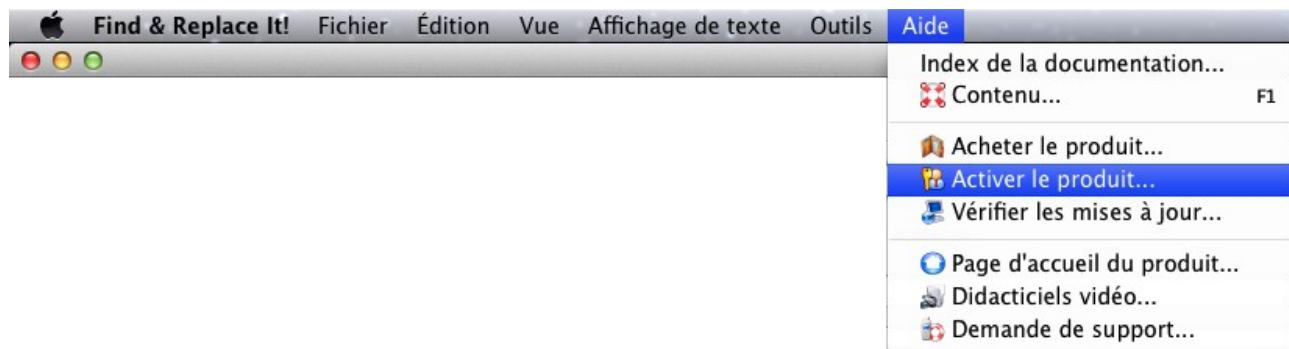
Once that the activation has been done there is no more possibility to use the demo version.

Note that if the field named **Valid through** is empty, then your license will last forever; otherwise, your activation will be revoked by the date mentioned in that field.

All activation keys refers to some hardware identifiers once they have been activated, hence it is not possible to use the same activation key on several computers. In the same way, a trial key is unique for a certain hardware and thus it is not possible to obtain several trial keys for the same computer.

Once the product has been activated, the activation dialog no longer shows up at startup time. However, this dialog might reappear from time to time if your hardware or software configuration has changed. In this case, you will simply have to reactivate your key. This will automatically update your hardware ID in our database. If too many changes occur, the activation might be refused. In such a case we kindly request you to [contact us](#) in order to reactivate your copy of the software.

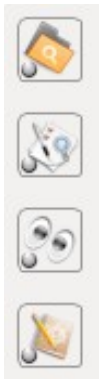
You can check your activation key at any time through the **Help/Activate product...** menu as shown below:



4 User Interface

The GUI of *Find & Replace It!* consists of ten panels that you can arrange in any way you want. You can select which component must be displayed as central widget; you can dock panels side by side, arrange them in tabs, or make them float. Adjust the windows layout to the way you like to work. This enables you to be at ease on a wide range of screen resolutions, from your laptop with its 13.3-inch display up to your 30-inch display.

Find & Replace It! comes with four pre-arrange layouts, one for each possible central widget, that can be customized the way you want depending on the size of your screen. You can switch from one layout to another in a simple click.



Four panels can take place in the center of the main window as a base for a pre-arranged layout:

- ***Files Selector***;
- ***Find & Replace Editor***;
- ***Find & Replace Preview***;
- ***Replace Command List***.

Each of these panels can become the center of a pre-arranged layout that can be customized. The left side bar of the main window provides a shortcut for toggling the visibility of any of these windows.

To have the ***Files Selector*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing **Ctrl+1**.



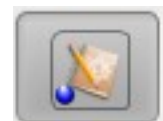
To have the ***Find & Replace Editor*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing **Ctrl+2**.



To have the ***Find & Replace Preview*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing **Ctrl+3**.



To have the ***Replace Command List*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing **Ctrl+4**.



This central widget can't be moved, but it can be resized (to become more or less large, and more or less high).

The top, bottom and right (not left) areas around the central widget are classical dock areas for dockable widgets.

To toggle the visibility of the windows that you want to have on the screen, you have to check mark some of the seven icons located on the left side bar of the main window. These seven icons are shown below:



Files Selector

Shortcut: **Ctrl+F1**



Find & Replace Preview

Shortcut: **Ctrl+F2**



HTML Viewer

Shortcut: **Ctrl+F3**



Find & Replace Editor

Shortcut: **Ctrl+F4**



Regular Expression Editor

Shortcut: **Ctrl+F5**



Replace Command List

Shortcut: **Ctrl+F6**



Encoding Conversion

Shortcut: **Ctrl+F7**



Find Text

Shortcut: **Ctrl+F8**



Output

Shortcut: **Ctrl+F9**



Find & Replace Preferences

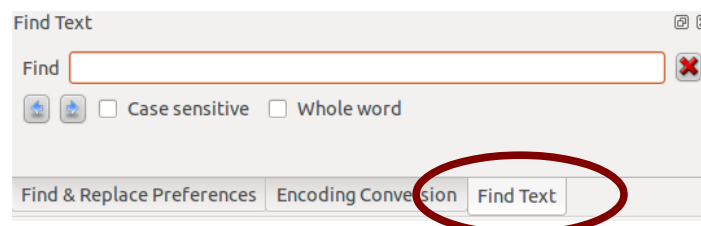
Shortcut: **Ctrl+F10**

Every panel (also called widget, window) can be floating, stacked one on the others or just docked.

As you can see, the three central widgets of the three pre-configured layouts are also available as dockable widgets, when one of the other two pre-configured layout is selected. In other words, when a particular window is displayed as a central widget, it is not any more available as a dockable window. This simply means you can't have twice the same window on the screen.

If you have two or more stacked widgets on the others in the same dock area, you can see them as tabs at the bottom of the area.

For example, when the **Find Text** is stacked over other dockable widgets you get that:



5 Two-Minute Guide to Replacing Text in Files

1. Show the Files Selector.

2. Select the folder in which to search for files.

3. Enter the appropriate filters to search for files.

4. Check mark all files to process (search, replace, convert encoding, etc.)

5. Open the Find & Replace Preferences in order to tell whether you want to backup modified files or not.

6. Show the Find & Replace Editor.

7. Enter the expression to search for.

8. Enter the expression to replace with.

10. Click on the appropriate button to start searching or replacing text in files.

9. Verify the matched expressions in the preview.

11. Show the Output window.

12. Verify the statistics on texts found or replaced.

The top screenshot shows the 'Find scope' window. It has a 'Search in' field with a file path, a 'File name filter' with a wildcard pattern, and a 'File path exclusion filter' with a list of paths. A 'Found files' table lists files with checkboxes for selection. The bottom screenshot shows the 'Find & Replace Editor' with 'Find' and 'Replace' fields, and a 'Find & Replace Preview' window showing the results of the search and replacement. The 'Output' window at the bottom shows the statistics of the search and replacement process.

6 Functions

6.1 Selecting Files to Process

The files to be searched and / or converted can be selected through the panel called **Files Selector**.

There are two sections that work together for searching and selecting files. They are **Search scope** and **Found files**.

Search scope

Search in

☒ Search in subfolders ☐ Include hidden and system folders

File name filter



	File path exclusion filter	Filter syntax	Case sensitive
1	/3rdparty/	Simple text	<input type="checkbox"/>
2	/temp/	Simple text	<input type="checkbox"/>
3	/ui_*	Wildcard	<input checked="" type="checkbox"/>


Found files

Filter Syntax ☒ Simple text

☒ Case sensitive filter ☒ Case sensitive sort

	File path	Encoding	BOM	End-of-line	R/W	Size	Modified
17	<input type="checkbox"/> /home/common/dev/ezconnect/src/gui/PMainFrm.h	UTF-8	None	Windows	R/W	4 KB	18.07.09 15:50
18	<input type="checkbox"/> /home/common/dev/ezconnect/src/twofish/AES.H	ISO-8859-1	None	Windows	R/W	10 KB	25.01.09 22:12
19	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/twofish/DEBUG.H	ISO-8859-1	None	Windows	R/W	2 KB	11.01.09 05:23
20	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/twofish/PLATFORM.H	ISO-8859-1	None	Windows	R/W	2 KB	11.01.09 05:23
21	<input type="checkbox"/> /home/common/dev/ezconnect...wofish/PTwofishWrapper.cpp	UTF-8	None	Windows	R/W	9 KB	18.07.09 15:50
22	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/twofish/PTwofishWrapper.h	UTF-8	None	Windows	R/W	2 KB	18.07.09 15:50

1. Select a folder to search in. This is the root path where you want to scan for files. You can type a path directly in the **Search in** text field or use the button on the right.
2. Choose to search files recursively into sub folders or not by toggling the **Search in sub-folders** check mark. When searching recursively you can choose to include the hidden folders or not with the **Include hidden and system folders** check box.
3. Enter zero, one or many file name filters within the **File name filter** field. These filters interpret [wildcard](#) characters like '*'. They must be comma separated.
4. Optionally add one or more expressions to exclude some file paths when searching for files. This can be achieved with the  button. These filters can use [wildcard](#) or [regular expression](#) syntax. Note that all file paths are described with '/' separator whatever the platform or system locale is. It is possible to remove filters by selecting the appropriate rows and then clicking on .
5. Select files that you want to process in the found files list. Unmarked files are not going to be read or touched. The content of this list is updated whenever you change search options. You can filter the content of this list through the file path filter above the list view. Click the column header to sort that column.

The scan of the hard-drive for searching files is a pretty long operation when running on a very large directory structure, however it is possible to stop it at any time by clicking on  or by pressing the **Escape** key.

The column labelled **R/W** indicates the file permissions for the reading or writing. If the reading is not allowed on a file, it cannot be selected and consequently it cannot be processed. In that case the permissions appear on a red background.

File path	Encoding	BOM	End-of-line	R/W	Size	Modified
lector/fil...tor/PFilePathFilterProxyModel.cpp	ISO-8859-1	None	Unix	R/-	14...es	18.09.10 13:41
lector/file...lector/PFilePathFilterProxyModel.h	Unknown	None	Undefined	-/W		18.09.10 13:41
lector/fileselector/PFileSelectorFrm.cpp	Unknown	None	Undefined	-/-	3 KB	17.04.09 20:10
lector/fileselector/PFileSelectorFrm.h	Unknown	None	Unix	R/W	80...es	18.09.10 13:41
lector/fi...r/debug/moc_PFileSelectorFrm.cpp	ISO-8859-1	None	Unix	R/W	3 KB	17.04.09 20:02
lector/fileselector/main.cpp	Unknown	None	Undefined	-/-	18...es	16.04.09 07:56
lector/fi...ctor/qt-gdbmacros/gdbmacros.cpp	ISO-8859-1	None	Unix	R/W	77 KB	17.04.09 20:02
lector/fileselector/ui_PFileSelectorFrm.h	Unknown	None	Unix	R/W	15 KB	18.09.10 13:41

6.2 Detecting or Selecting the Encoding of Files

Character set detection is the process of determining the character set, or encoding, of character data in an unknown format. This is, at best, an imprecise operation using statistics and heuristics. Because of this, detection works best if you supply at least a few hundred bytes of character data that's mostly in a single language. In some cases, the language can be determined along with the encoding.

Several different techniques are used for character set detection. For multi-byte encodings, the sequence of bytes is checked for legal patterns. The detected characters are also check against a list of frequently used characters in that encoding. For single byte encodings, the data is checked against a list of the most commonly occurring three letter groups for each language that can be written using that encoding. The detection process is configured to ignore html or xml style markup, which can interfere with the detection process by changing the statistics.

The **Found files** section shows the found files according to your current options for searching files. The column **Encoding** is the only one that is editable by the user. You can choose by hand the appropriate codec for each file with a click on the appropriate row, in the **Encoding** column. This will show up a drop-down list of available codecs as shown below:

	File path	Encoding	BOM	End-of-line	R/W	Size	Modified	Created
8	<input checked="" type="checkbox"/> /home/common/dev...index-local.html	UTF-8	None	Unix	R/W	1 KB	21.07.09 18:08	11.09.10 10:47
9	<input checked="" type="checkbox"/> /home/common/de...mple/index.html	UTF-8	None	Unix	R/W	1 KB	23.06.10 08:07	11.09.10 10:47
10	<input checked="" type="checkbox"/> /home/common/de...mple/style.css	ISO-8859-1	None	Unix	R/W	53...es	21.07.09 18:08	11.09.10 10:47
11	<input checked="" type="checkbox"/> /home/common/de...tes/product.php	ISO-8859-1					10 08:07	11.09.10 10:47



The “best codec” for processing text encoding of each file is detected using statistics and heuristics, and it is selected by default. Please note that these heuristics are only reliable for detecting Unicode charsets. For other encodings it will only give you some suggestions. There might are many “acceptable codecs” and they are all marked with a blue light on the left side of the drop-down list. This is shown below:

Found files
Filter: components
☐ Case sensitive filter ☒ Case sensitive sort

	File path	Encoding
98	<input checked="" type="checkbox"/> /home/philippe/dev/...in.content.html.php	
99	<input checked="" type="checkbox"/> /home/philippe/dev/...t/admin.content.php	
100	<input checked="" type="checkbox"/> /home/philippe/dev/..._content/config.xml	
101	<input checked="" type="checkbox"/> /home/philippe/dev/...content/content.xml	
102	<input checked="" type="checkbox"/> /home/philippe/dev/d...ntent/controller.php	

- ISO-2022-JP (JIS7)
- ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)
- ISO-8859-10 (iso-ir-157, latin10, ISO-8859-10, ISO-8859-10, csISOLatin10)
- ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)
- ISO-8859-13
- ISO-8859-14 (iso-ir-199, latin8, iso-celtic)
- ISO-8859-15 (latin9)
- ISO-8859-16 (iso-ir-226, latin10)
- ISO-8859-2 (latin2, iso-ir-101, csISOLatin2)
- ISO-8859-3 (latin3, iso-ir-109, csISOLatin3)
- ISO-8859-4 (latin4, iso-ir-110, csISOLatin4)
- ISO-8859-5 (cyrillic, iso-ir-144, csISOLatinCyrillic)

If the codec name is set to **Unknown** for a given row (and a gray light appears beside the codec name), that means that no codec seems to be consistent with the associated file content. When one or more codecs are detected as acceptable, the preferred text encoding is selected by default when it is available. The list of acceptable codecs is automatically determined whenever a new file is displayed in the list but the list is not updated when files are changed on your hard-drive.

For refreshing the encoding detection, click on . This will detect the acceptable codecs for all found files listed and clear any previous codec selection made manually by the user. The detection process will attempt to identify the charset that best matches the characteristics of the byte data, but the process is partly statistical in nature, and the results can not be guaranteed to always be correct. For best accuracy in charset detection, the input data should be primarily in a single language, and a minimum of a few hundred bytes worth of plain text in the language are needed. The detection process will attempt to ignore html or xml style markup that could otherwise obscure the content. The charset detection is a pretty long operation when running on thousands of files, however it is possible to stop it at any time by clicking on  or by pressing the **Escape** key. If you need detecting the encoding on a few files only, you might use the command named **Detect files encoding for selection** in order to avoid performing the detection on thousands of files.

In order to increase performances of encoding detection as well as end-of-lines detection, all data are stored in a cache. That means that once a the text encoding (or line ending) has been detected, the detection does not occur anymore. These cached data are automatically refreshed whenever the file is modified. However, in case you really want to forcing a new detection, you can clear at any time the cached content with the command **Clear current files encoding cache**, located in the context menu of the **Found files** table view. The number of files for which the se encoding information are cached can be set in the [preferences](#) panel.

If you need to look at the decoded content of a file, right-click on a file entry, then click on [Open file in test preview](#) (or double-click on the appropriate row in the table). This will allow you to play with the codec used to decode the file.

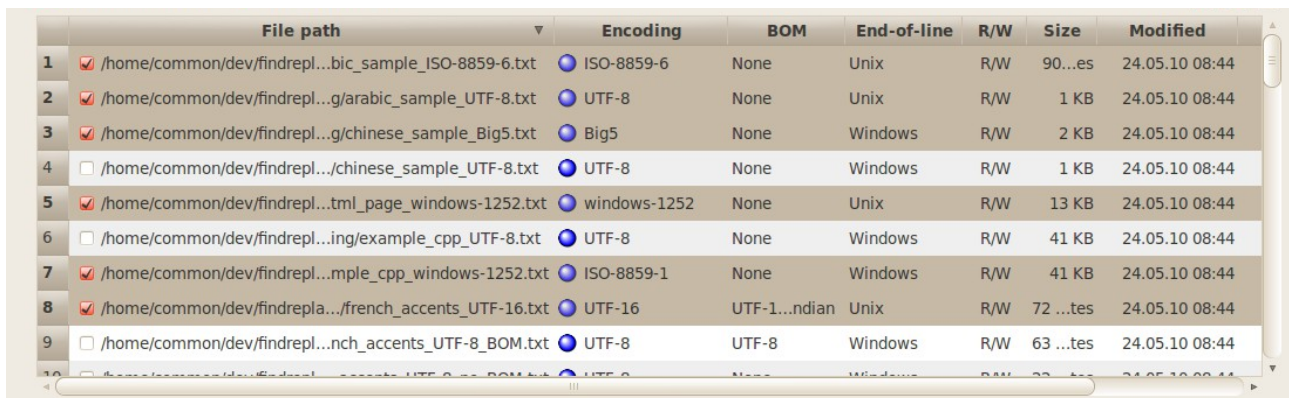
It is possible to open a file in an external editor straight from the list. For that, right-click on the appropriate row in the table, then click on **Open file in default editor** (or double-click on the appropriate row in the table with the **Shift** key pressed).

Note: Character set detection is at best an imprecise operation. The text-encoding detection uses heuristics that do not always provide accurate results except for all Unicode encodings (i.e.: UTF-8, UTF-16, UTF-32).

6.3 Converting the Encoding of Files


For converting the text encoding of a given set of files, follow these steps:

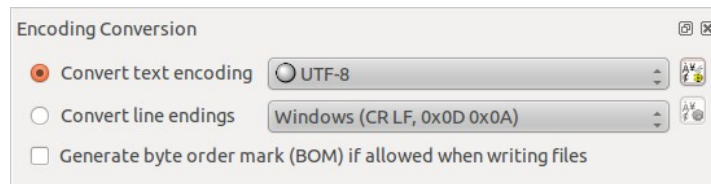
1. Select the files you want to convert with a check mark in the **File path** column, within the **Found files** section (see screenshot below);





	File path	Encoding	BOM	End-of-line	R/W	Size	Modified
1	<input checked="" type="checkbox"/> /home/common/dev/findrepl...bic_sample_ISO-8859-6.txt	ISO-8859-6	None	Unix	R/W	90...es	24.05.10 08:44
2	<input checked="" type="checkbox"/> /home/common/dev/findrepl...g/arabic_sample_UTF-8.txt	UTF-8	None	Unix	R/W	1 KB	24.05.10 08:44
3	<input checked="" type="checkbox"/> /home/common/dev/findrepl...g/chinese_sample_Big5.txt	Big5	None	Windows	R/W	2 KB	24.05.10 08:44
4	<input type="checkbox"/> /home/common/dev/findrepl.../chinese_sample_UTF-8.txt	UTF-8	None	Windows	R/W	1 KB	24.05.10 08:44
5	<input checked="" type="checkbox"/> /home/common/dev/findrepl...tml_page_windows-1252.txt	windows-1252	None	Unix	R/W	13 KB	24.05.10 08:44
6	<input type="checkbox"/> /home/common/dev/findrepl...ing/example_cpp_UTF-8.txt	UTF-8	None	Windows	R/W	41 KB	24.05.10 08:44
7	<input checked="" type="checkbox"/> /home/common/dev/findrepl...mple_cpp_windows-1252.txt	ISO-8859-1	None	Windows	R/W	41 KB	24.05.10 08:44
8	<input checked="" type="checkbox"/> /home/common/dev/findrepla.../french_accents_UTF-16.txt	UTF-16	UTF-1...ndian	Unix	R/W	72 ...tes	24.05.10 08:44
9	<input type="checkbox"/> /home/common/dev/findrepl...nch_accents_UTF-8_BOM.txt	UTF-8	UTF-8	Windows	R/W	63 ...tes	24.05.10 08:44
10	<input type="checkbox"/> /home/common/dev/findrepl...accents_UTF-8_BOM.txt	UTF-8	None	Windows	R/W	23 ...tes	24.05.10 08:44

2. [Select the current text encoding of these files](#) if the auto-detected encodings are not accurate;

- Open the **Encoding Conversion** panel by clicking the  button in the left side bar;

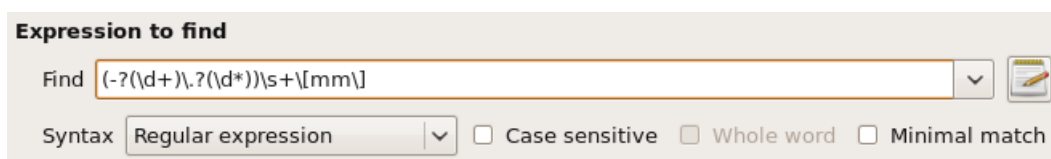



- Select the target encoding for your set of files;
- Optionally you can select the **Generate Byte Order Mark (BOM)** check box. This will insert the BOM (Byte Order Mark) at the beginning of the file when it is written. This option only apply to Unicode text encoding: UTF-8, UTF-16 and UTF-32. Note that this option may interfere with the target encoding. For instance, if you choose a Unicode encoding that does not allow the BOM, it will turn your target encoding to the closest Unicode encoding that allows it.
- Optionally you might schedule [a backup of modified files](#);
- Click on the button  to start the encoding conversion. If you want to stop the conversion process, click on the button . All converted files have a green light beside their codec if the conversion succeeded; otherwise the light is red. Note that refreshing the encoding detection will restore the classic blue or gray lights.

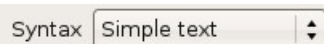
6.4 Searching for an Expression in Files

The **Find & Replace Editor** panel allows you to define what you want to search or replace. This is the place where you configure the string to find in files you want to search. Additionally this is also the tool you use to define the string you use to replace the strings found.

The **Expression to find** section allows you to setup an expression to search for:




- Enter the expression to search into the **Find** text field. Alternatively you might use the  button to edit your expression with the [Regular Expression Editor](#).
- You can choose the way your expression must be interpreted through the **Syntax** drop-down list:
 - [Simple text](#): each character is understood in its literal meaning.




- [Wildcard](#): is similar to the functionality found in command shells with wildcard characters that can be used as a substitute for any occurrences of a class of characters.
 - Wildcard Unix: this is similar to Wildcard but with the behavior of a Unix shell. The wildcard characters can be escaped with the character '\\'.
 - [Regular expression](#): is a sequence of characters that forms a search pattern which is called regular expression statement. That statement is expressed in terms of a grammar in the formal language supported by the Qt's regular expression processor.
- Select the options that apply when matching against your expression:



The 'Minimal match' match option is only available when 'Regular expression' syntax is set. This turns the [quantifiers](#) in non-greedy mode.

- [Select the files you want to scan](#) with a check mark in the **File path** column;
- Select the current text encoding of these files if the auto-detected encodings are not accurate;
- Optionally you can test your expression with the [Find & Replace Preview](#);
- Click on the  button to start searching your expression into the [selected files](#). If you need

to stop the search, click again on the same button which has been morphed into  once the search has been started.

Note: For performance reasons it is not possible to search for an expression longer than 50'000'000 characters. By default this limit is fixed to 100'000 to increase search speed. This can be changed through the [Find & replace preferences](#) panel.



6.5 Replacing a Given Expression in Files

The **Find & Replace Editor** allows you to define what the strings to search and replace in files.

1. [Setup the expression](#) you want to search;
2. Type your replacement pattern for matched occurrences of your expression in the **Replace with** text field:

☒ Replacement expression

Replace with 

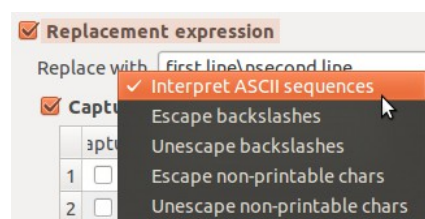
3. Optionally you might schedule [a backup of modified files](#);
4. Click on the  button to start replacing your expression into the [selected files](#). If you need to stop the replacement, click again on the same button which has been morphed into  once the replacement has been started.

It is possible to interpret ASCII sequences given in replacement text. An ASCII sequence is set of printable chars that are interpreted as a single non-printable char. This is useful whenever you want to insert non-printable chars into your replacement text (e.g: line feed, carriage return, tab, etc.). The following table shows the supported ASCII sequences:

ASCII Sequence	Non-printable char
\a	Bell
\b	Backspace
\f	Form feed
\n	Line feed
\r	Carriage return
\t	Horizontal Tab
\v	Vertical Tab
\\	Literal backslash

Any sequence starting with the escape char “\” that does not figure in this table is ignored.


The drawback of ASCII sequences is that you need to escape “\” to “\\” to get a literal “\”. Since it might be cumbersome to escape every “\”, especially when you manipulate file paths, ASCII sequences are not interpreted by default. To turn it on you must do a right-click on the label **Replace with** and select Interpret ASCII sequences. Note that it is possible to type non-printable chars in the replace pattern text box (e.g: copy & paste a tab char from an existing document).

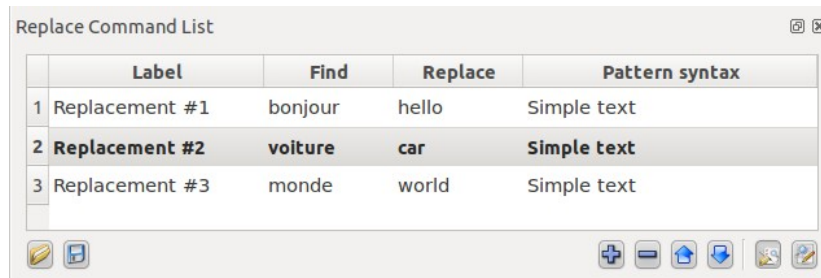


The ASCII sequence is turned on. That is mandatory to grant access to the other command of the context menu.

The context menu of the label **Replace with** offers convenient tools for escaping non-printable chars as well as backslashes.



6.6 Executing Several Replacements in One Shot



The window **Replace Command List** lets you define a sequence of distinct replace operations. Once that the replacement sequence is defined it can be executed in one click with the command **Replace in files** which is accessible via the button .





A sequence composed of three replace commands. The second row appears in bold. That means the second command is in edition mode.

Each row represents a find & replace "command" that you can edit with the [Find & Replace Editor](#). Each command is identified by a label that can be edited by pressing **F2** or double-clicking in the label's cell.

The button  adds a new row (i.e. a new replace command) to the list below the selected line. The command is added from the current contents of the **Find & Replace Editor**. The button  removes all selected rows from the list (i.e. removes the selected replace commands).

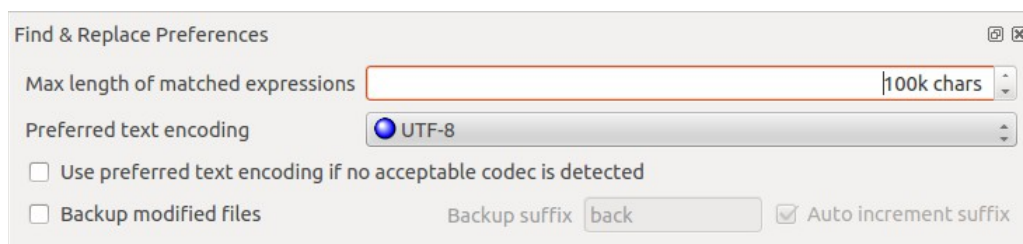
Each replace command can be edited at any time by either double-clicking on a row (except in the label column) or using the button . Be aware that entering the edition mode will replace the current contents of the **Find & Replace Editor** with the contents of the command to edit. When a command is in edition mode, the font of the edited row is in bold. In edition mode, each modification made in the **Find & Replace Editor** is reflected in the edited command row. You can end the edition by toggling the state of button  or by selecting any other row in the list.

You can modify the order of the sequence by selecting the rows you want to move and using the move buttons  and .

6.7 Editing Find & Replace Preferences

The **Find & Replace Preferences** panel offers some control on default behaviors for searching text, detecting encodings or doing backups of files.

The **Find & Replace Preferences** panel is accessible by clicking the  button in the left side bar.



The **Max length of matched expressions** field indicates the maximum number of chars that can be matched by a regexp. It is not possible to find or replace a string that is longer than this value. This value implicitly defines the size of the cache used to process the contents of files. This value

implicitly defines the size of the cache used to process the contents of files. A large value implies working with a large cache. Since the size of the cache greatly impacts the performance of the processing (depending on the situation), it is strongly recommended not to increase too much this value. Indeed, in order to speed up numerous replacements in small amounts of data, it's recommended to decrease the default value (assuming you don't need to replace huge strings). As an example, if you replace every char in a file, one by one, with a string consisting of twice the found letter it's much faster to have the smallest possible size of cache. Conversely, it's much faster to have a very big cache size when you only search strings without replacing them.

6.7.1 Changing the Encoding Detection Behavior

The **Preferred text encoding** field tells the system which encoding to choose by default when there is more than one compatible codec detected. Of course, if the encoding detection evaluates that the encoding is invalid for a given file, it will not be forced.

The option **Use preferred text encoding if no acceptable codec is detected** forces the **preferred text encoding** whenever no acceptable encoding can be automatically proposed. Since version 2.0.8, this option does not force the preferred codec anymore for binary files.

6.7.2 Doing Backups of Files before Modification

The **File backup** section allows you to create a backup for each modified files.

1. Mark the **Backup modified files** check box as shown above;
2. Type a suffix for your backup files into the **Backup suffix** edit. If a file with the same name already exists, it will be replaced. So you should consider to update the file suffix between each manipulation of the original file;
3. **Auto increment suffix**: Optionally you may select this option for incrementing the suffix so that each backup will have a unique name (e.g. `<a-file>.back.001`, `<a-file>.back.002`, etc.). Otherwise, each backup file overwrites the previous one.

6.8 Using the Find & Replace Preview

When editing an expression to search for, it is convenient to match it against real data. In order to achieve this, let's take a tour of the **Find & Replace Preview** window.

With the preview window you can visualize in one glance what are the strings that match a given expression, what will be the replacement text, and even mix both views. It gives you an immediate feed-back on what is going to change, which makes easy to understand the impact of your replacement within a file without actually modifying its content. Such a realtime visualization is especially useful whenever you want to build a complex expression to search for. The preview can also be used as a multi-document editor.



Here are the key features of this tool:



- check the impact of a given text encoding when applied to a file content;
- edit a text sample against which you want to match an expression to find;
- preview matched occurrences of an expression to find inside a given text sample;
- preview resulting content of a text sample after the replacement of all occurrences of your expression with your replacement pattern;
- preview both found expressions and processed replacements inside a text sample;
- navigate through found occurrences of a specific expression or replacement pattern;
- WYSIWYG preview for [HTML documents](#).


All these features are of serious help to search and replace strings in multiple files without harm. Indeed, changing many files using a find and replace strategy, might lead to severe damages if


your not a little cautious. Of course, it's likely you're working with a versioning system, but there are still some situations where you might quickly lose some contents, or simply corrupt your files.

6.8.1 Previewing as Plain Text

You can open files in the preview by clicking on . The button  allows saving the content of the preview. The name displayed in the tab will be used as the file name for saving. If the label of the tab is not a valid file path you will have to select a destination file. You can force to select a destination file path by using the drop-down arrow beside the button, and select **Save as....**

The drop-down button  lets you start a new preview document, or use the contents of the active preview as source text for a new preview ().

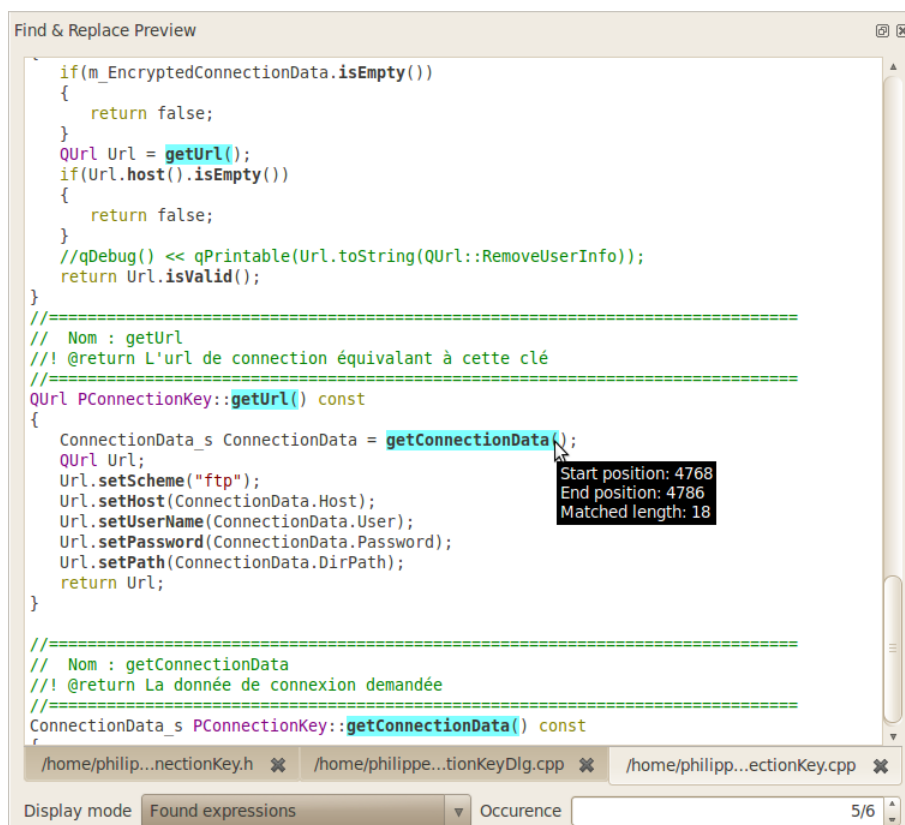
The drop-down button  lets you clear the contents of the active preview, close the current preview tab, or close all tabs.

It is possible to preselect the preferred codec for the preview of upcoming files. In fact, the active codec of the preview is used as preferred codec for the next opened file. It is also possible to reload the content of the file in the preview with the button . Thus the selected codec will be used to read the file.

It is possible to copy the file path of the active preview to the clipboard by right-clicking on the tab bar. In the same way it is also possible to open the active document within the default editor associated to this type of file.

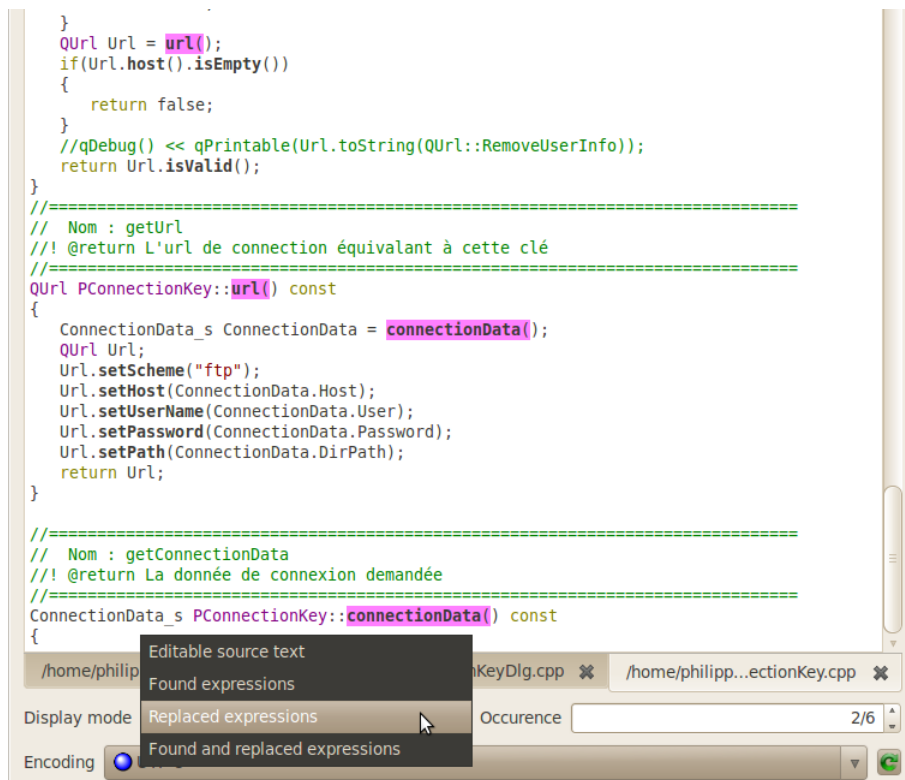
Note that all line breaks in the preview are internally represented by a Line Feed character (LF, U+00A). This is always true, whatever the original end-of-line used in the displayed file. If you want to search for a multi-line expression with another style of line break, we strongly advise you to use a regular expression with appropriate `\s+` sequences in order to match any kind of end-of-line.

The following screenshots illustrate some of the capabilities described above.

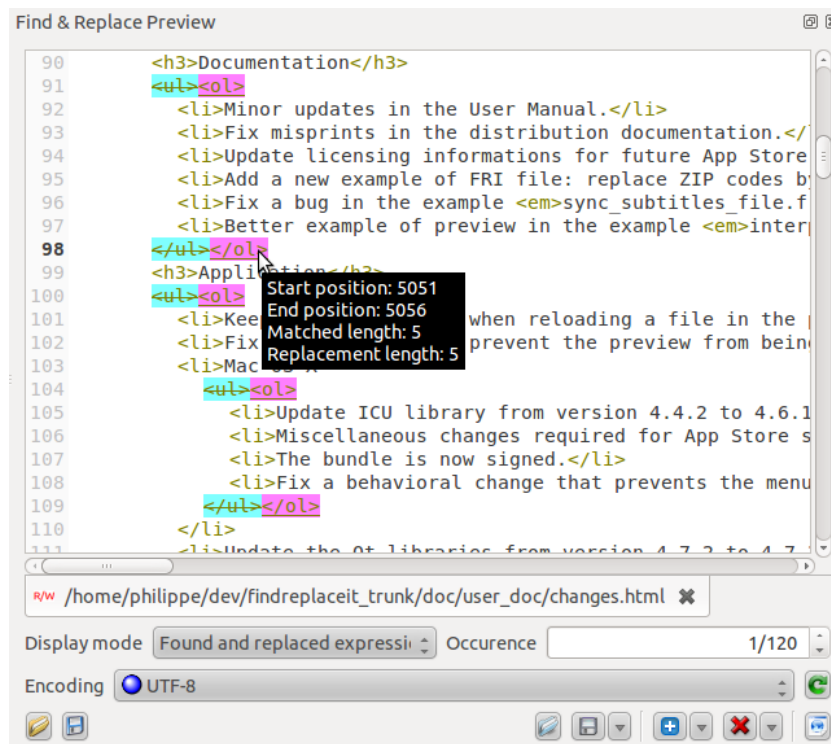


Highlight of matched expressions within a file. The tool-tip shows information about occurrence location.

The drop-down list named **Display mode** allows to switch the content displayed into the active preview editor. This feature is demonstrated into the two following screenshots:




Highlight of replaced expressions within a file.

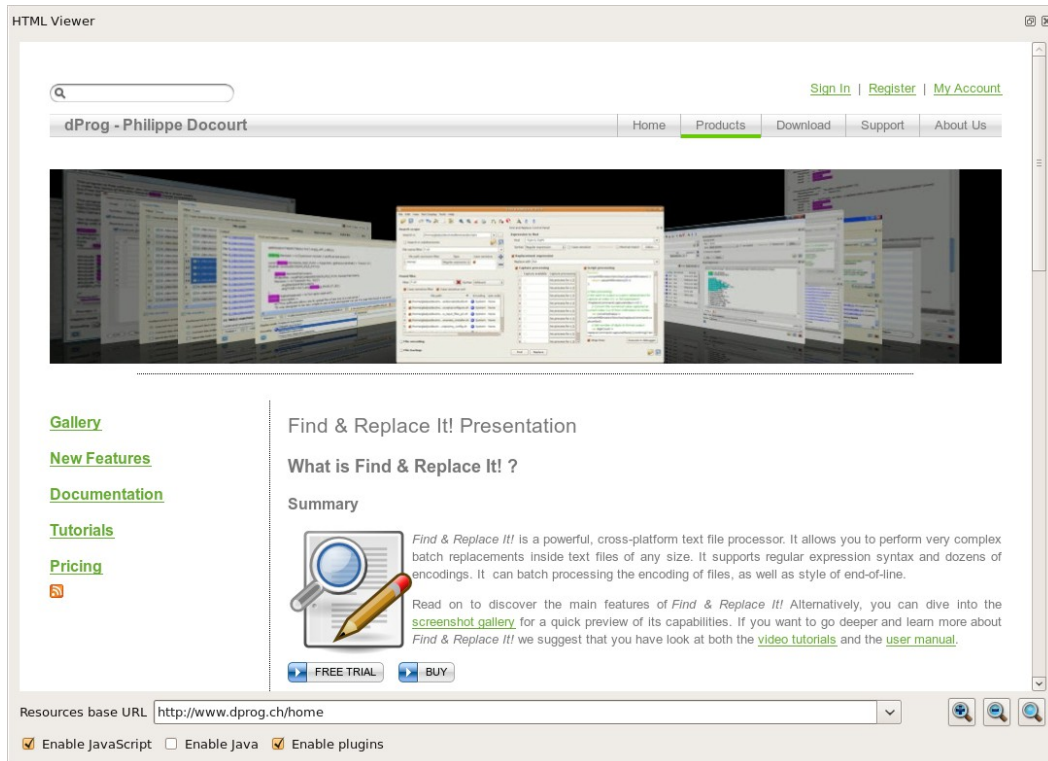


Highlight of both found and replaced expressions within a file. The tool-tip shows information about replacement location.

Note: For performance reasons the **Find & Replace Preview** has a content limit of 100'000'000 characters. By default this limit is fixed to 5'000'000 to increase search speed. This can be changed through the [preferences panel](#).

6.8.2 Previewing as HTML

In addition to the plain text preview, you can activate the **HTML Viewer** through the  button. That will enable you to preview HTML documents, with either their original or altered content, without having to save them.



The HTML preview in action with our own website.

The viewer is directly synchronized with the current content of [plain text preview](#). That means the **Display mode** will also apply on the HTML content.

Because the viewer cannot resolve relative links in the HTML document from the content of the preview, you might need to enter an appropriate URL. This URL is used to process all resources referenced by relative links within the document (i.e.: CSS, images, scripts, etc.). This is not required when all resources are given with absolute path.

The **HTML Viewer** provides rendering of HyperText Markup Language (HTML), Extensible HyperText Markup Language (XHTML) and Scalable Vector Graphics (SVG) documents, styled using Cascading Style Sheets (CSS) and scripted with JavaScript. Some common plugins are also supported through the Netscape Plugin API, provided you have appropriate binary files for those plugins installed on your computer. The following locations are searched for plugins:

Linux/Unix

- `.mozilla/plugins` in the user's home directory
- `.netscape/plugins` in the user's home directory
- System locations, such as
 - `/usr/lib/browser/plugins`
 - `/usr/local/lib/mozilla/plugins`
 - `/usr/lib/firefox/plugins`
 - `/usr/lib64/browser-plugins`
 - `/usr/lib/browser-plugins`
 - `/usr/lib/mozilla/plugins`
 - `/usr/local/netscape/plugins`

Linux/Unix

- /opt/mozilla/plugins
- /opt/mozilla/lib/plugins
- /opt/netscape/plugins
- /opt/netscape/communicator/plugins
- /usr/lib/netscape/plugins
- /usr/lib/netscape/plugins-libc5
- /usr/lib/netscape/plugins-libc6
- /usr/lib64/netscape/plugins
- /usr/lib64/mozilla/plugins
- Locations specified by environment variables:
 - \$MOZILLA_HOME/plugins
 - \$MOZ_PLUGIN_PATH
 - \$QTWEBKIT_PLUGIN_PATH

Windows

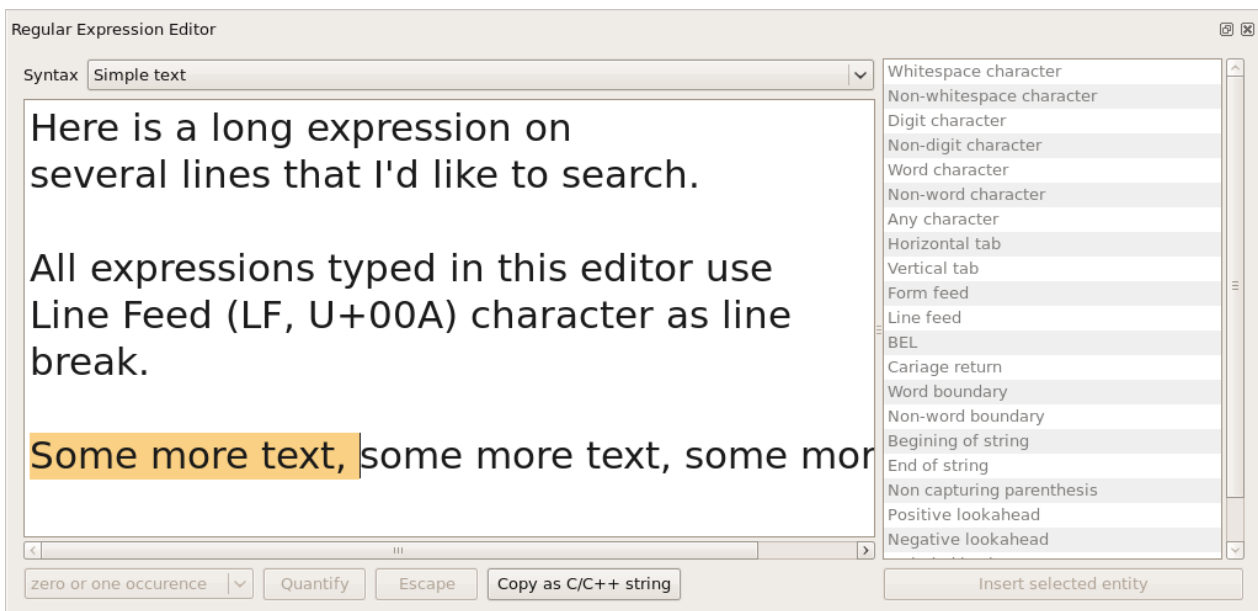
- The user's Application Data\Mozilla\plugins directory
- Standard system locations of plugins for Quicktime, Flash, etc.

Mac OS X

- Library/Internet Plug-Ins in the user's home directory
- The system /Library/Internet Plug-Ins directory

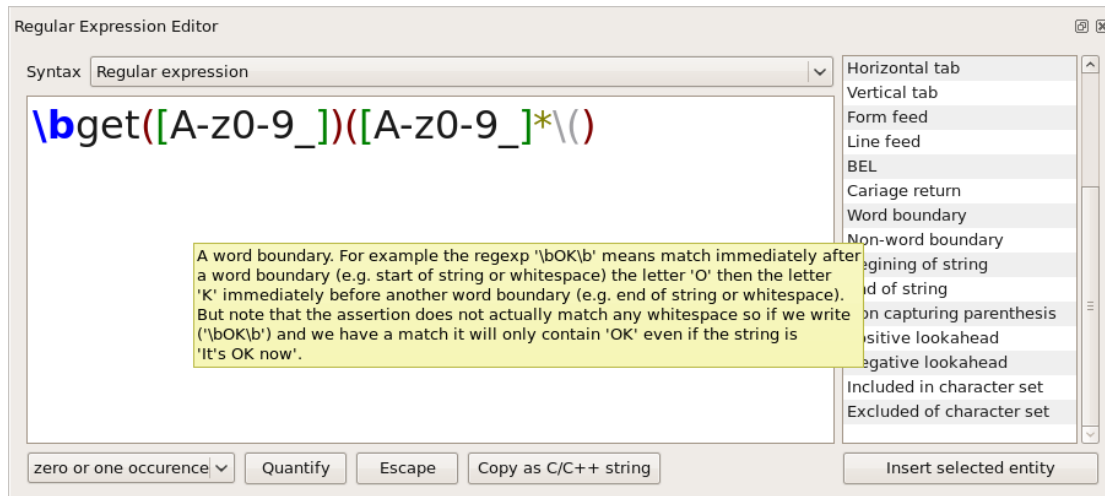
6.9 Using the Regular Expression Editor

When you need to write a multi-line expression to search, the **Regular Expression Editor** is your best friend. The multi-line edition is shown bellow with a **simple text** expression:



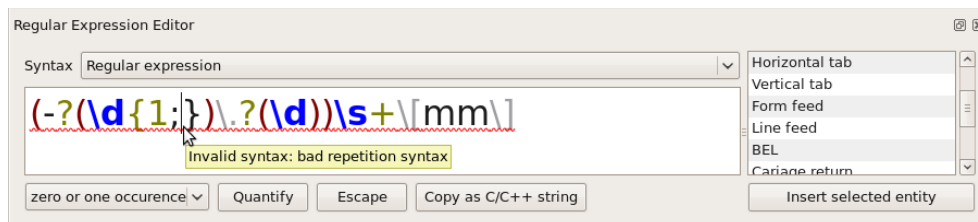
Note that all line break are internally represented by a Line Feed character (LF, U+00A). If you want to search for a multi-line expression with another style of line break, we strongly advise you to use a regular expression with appropriate '\s+' sequences in order to match any kind of end-of-line.

In addition, this editor simplifies the setup of regular expression. It provides tools to manage regular expression entities. On the right side of the text editor, there is a list of available regular expression entities (e.g.: special characters, grouping expression, etc.). If you leave the mouse over an item of this list, an explanatory tooltip will appear. This is shown in the figure below:



The Regular Expression Editor has some nice features like syntax highlighting and scope matching (e.g.: matching scope for '()', '[]', '{}').

The **Regular Expression Editor** offers an automatic syntax check for wildcard and regular expressions:



As soon as your wildcard or regular expression pattern becomes invalid, it is underlined. A tooltip provides a brief description of the syntax error detected.

6.10 Advanced Replacements

Advanced replacement covers three main features that makes *Find & Replace It!* really powerful:

- Injecting a fragment of the matched expression into the replacement text;
- Transforming a fragment of the matched expression before injecting it into the replacement text;
- Transforming any text that is going to be replaced by using a JavaScript interface.

Each of these points is described in the following chapters.

6.10.1 Using of Captured Texts within Replaced Texts

This feature requires regular expression syntax for the expression to find; furthermore, you should be familiar with captures within regular expressions. To learn more about these notions we recommend to read the [Regular Expressions chapter](#). If you are familiar with regexp, read on the following example.

Whenever you capture some text fragments with an expression to find,

Find

you can inject these captured fragments into your replacement pattern. This is done with a `%1`, `%2`,

..., %9 pattern. Where the number that follows the percent sign is the capture index. %0 is a special, implicit capture that includes the full expression match.

Replace with

Let's imagine we have a CSV file containing contacts like in the following snippet:

```
First Name: John; Family Name: Smith; Phone: ...
First Name: Mike; Family Name: Dupont; Phone: ...
```

We want to swap the first two columns. Here we have to capture two variable expressions (first name and family name) and move them around. Here is an easy way to do it.

Find:

(First Name: [^;]+); (Family Name: [^;]+)

The parentheses in expression above will capture two fragments of every matched occurrences in the CSV file.

Replace with:

%2; %1

The replacement pattern above is a dynamic text that varies for every matched occurrences. In fact %1 will be replaced by the content matched by the first parentheses scope. Idem with %2 and the second parentheses scope.

If you use %n where n is greater than the number of captures, then @{__INJECTION__}#n will be inserted in the replaced text. If you need to replace some text with a literal %n in a situation where you captured some texts with your regexp, you'll need escaping the %n sequence. This can easily be done by adding a % sign in front of the sequence. Note that the escapement only applies in front of a digit (i.e.: 0 to 9). Everywhere else the escapement is not processed. Here are some examples:

Text to process	Find	Replace with	Result
ab cd	(\w+)	%1	ab cd
56 78	(\d+)	%1%2	56@{__INJECTION__}#2 78@{__INJECTION__}#2
56 78	(\d+)	%1%%2	56%2 78%2
56	\d+	%%1	%1
56	(\d+)	%%1	%1
56 78	(\d+)	%%%1	%56 %78
56 78	(\d+)	%%%%1	%%1 %%1
56 78	(\d+)	%%%%2	%%2 %%2
56 78	(\d+)	%%%%%1	%%56 %%78
56 78	(\d+)	%%%%%2	%%@{__INJECTION__}#2 %%@{__INJECTION__}#2
ab	(\w+)	%z %1	%z ab
ab	(\w+)	%%z %%1	%%z %1
ab	(\w+)	%%%z %%%1	%%%z %ab

In the examples above, the pattern \w+ matches any sequence of one or more letters or digits. The same applies to digits only with the pattern \d+. These examples assume that the regexp is not set to "minimal match". Please read the [Regular Expressions chapter](#) for more details on the regexp syntax.

6.10.2 Processing the Captured Texts with Built-In Functions

This feature requires regular expression syntax for the expression to find; furthermore, you should be familiar with [captures](#) within regular expressions. To learn more about these notions we recommend to read the [Regular expressions chapter](#). If you are familiar with regexp, read on the following example.

As shown in the [previous chapter](#), it is possible to inject captured texts into your final replacement string, through a special syntax. It is also possible to apply an additional processing to captured strings before injecting them as a replacement expression. This can be handle with the **Capture processing** section:

	Capture available	Capture processing
1	<input checked="" type="checkbox"/>	To lower case
2	<input checked="" type="checkbox"/>	Apply script
3	<input type="checkbox"/>	No process for capture

Capture #1 and #2 are available but not capture #3. A distinct process has been attached to each capture, however it is not compulsory.

The left column is not editable. The check mark is toggled depending on the presence of captures and placeholders respectively within the expression to find and within the replacement pattern. The right column let you choose a transformation to apply to the capture, before injecting it as the replacement text at its placeholder location.

Let's imagine that we want to upper case the first letter that follows a ':' sign inside a file. A tedious solution might be to replace all : a with : A, : b with : B and so on. This will take some time. And then, what happens with accentuated letters or oriental characters? What happens if a tab sometimes replaces the whitespace after the ':' sign? What if there no whitespace at all or many white-spaces due to a typing mistake? This solution is definitively inappropriate. Here is a better way to handle this task:

Find:

```
: \s*(\w)
```

The expression above will match all ':' followed by any number of whitespace characters (including tabs and line breaks) and at least a word character.

In the **Capture processing** section, select **To upper case** as process for the first capture. This will transform to upper case the content matched by the first pair of parentheses, before injecting it as %1 in the replacement pattern.

Finally we replace with:

```
: %1
```

If the built-in capture processes are not sufficient, you might try the [scripting interface](#).

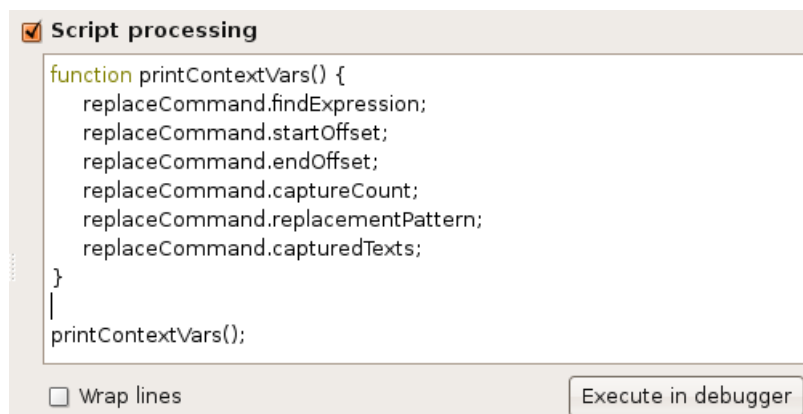
6.10.3 Processing the Replaced Texts by Script

Find & Replace It! has the ability to dynamically transform the texts to be replaced. This involves mixing all the classical possibilities of regular expressions with JavaScript programming. It is possible to look for very complex patterns with regular expressions. However a simple replacement by a static text is most of the time insufficient. Hopefully, *Find & Replace It!* provides a JavaScript-like interface to customize replacements on the fly. The scripting feature gives all the power of JavaScript to format whatever you need and the regular expressions simply select where to apply the script. This feature especially useful whenever you need some logic to interpret an expression and generate a replacement pattern based on it (e.g. find all numbers in a text and divide them by a given factor to convert units).

One of the most powerful feature of *Find & Replace It!* is its ability to dynamically transform the texts to be replaced. This involves mixing all the classical possibilities of regular expressions with JavaScript programming. It is possible to look for very complex patterns with regular expressions. However a simple replacement by a static text is most of the time insufficient. Therefore, the scripting feature gives us all the power of JavaScript to format whatever we need and the regular expressions simply tell us where to apply the script.

To carry out any of the mentioned examples above you only have to write an appropriate script in the script editor. Your script can be called at different occasions according to your wishes:

- Once with the original replacement pattern as **captureText** (see the table below) and the full expression match as **replacementText**. The property **captureIndex** equals 0. This call is automatic if the check box **Script processing** is ticked (see screenshot below).
- Once for each [captured text](#) in your expression if the related capture process has been set to **Apply script**. For each call, the **captureText** is set to the current capture, and the **replacementText** represents the replacement text after having been processed by all preceding capture processes. The property **captureIndex** gives the index of the current capture (from 1 to **captureCount**). This call only occurs if the check box **Script processing** is ticked.
- Once at the very end of the process, after all capture processing, with the resulting replacement pattern as **replacementText**, and an empty string as **captureText**. The property **captureIndex** equals **captureCount+1**. This call is automatic if the check box **Script processing** is ticked.



A dummy script outputting invariable properties from the [scripting context](#).

The scripting interface provides a simple way to access the context of the current match as well as the replacement pattern through the global **replaceCommand** object. The table below summarizes the context information made available to the script:

Properties of replaceCommand	Object or Data Type	Description
<i>Invariable properties: these variables do not change during the process of a given matched expression</i>		
findExpression	RegExp	Matched regular expression.
startOffset	Number	Starting offset of matched expression in the full text.
endOffset	Number	Ending offset of matched expression in the full text.
captureCount	Number	Number of captures contained in the expression to find.
capturedTexts	Array	Array of captured strings within the full expression match.

<i>Variable properties: these variables vary depending on the capture for which the script is called</i>		
captureIndex	Number	Index of the current capture for which the script has been called. This index is included between 0 and 'captureCount + 1'.
captureText	String	Captured text at current capture index. This property returns the full expression match when captureIndex is zero and returns an empty string when captureIndex equals 'captureCount + 1'. Otherwise it returns the captured string at the given index starting from (from 1 to captureCount).
replacementLength	Number	Length of the current replacement text. This length takes in account already replaced placeholders (i.e. %i).
replacementText	String	Current replacement text. This text contains the content of already replaced placeholders (i.e. %i). All placeholders that have not been replaced at the time of accessing this property (i.e.: i>captureIndex) are internally represented by @{__INJECTION__}#i. All escaped sequences of % are already treated in the text returned by this property.

The scripting interface also provides a simple way to access some properties of the file being parsed. You can read these properties through the global **currentFile** object. The table below summarizes the context information made available to the script:

Properties of currentFile	Object or Data Type	Description
<i>Invariable properties: these variables do not change during the process of a given matched expression</i>		
filePath	String	Full file path (absolute) including the name of the file.
dirPath	String	Absolute directory path where the file is located.
fileName	String	Full file name, including the extension.

Let's imagine we have a text file containing numerous numerical values. All of these values represent a length given in millimeters. We would like to convert all these distances from millimeters to inches. Sounds tricky to you? Script processing enables you to handle that task like any other replacement task. First we setup an expression that matches all numbers (integers and floating point):

```
(-?\d+\.\?(\d*))
```

As a replacement we simply inject the [full captured number](#) (the first capture in this example) by using the %1 placeholder in the replacement pattern:

```
%1
```

We might as well do this by using the special %0 placeholder, which represents the full expression matched by the regexp. By doing so, the external parentheses could have been omitted:

```
-?\d+\.\?(\d*)
```

As a replacement we simply inject the [full captured number](#) (the complete match in this example) by using the %0 placeholder in the replacement pattern:

```
%0
```

From now on we assume that we use the first example to search and replace for numerical values.

In order to apply some script on the replacement pattern activate capture processing (i. e. check mark), set process to **Apply script**, then activate the script processing editor option as shown below:

Capture processing		Script processing
	Capture available	Capture processing
1	<input checked="" type="checkbox"/>	Apply script
2	<input type="checkbox"/>	No process for capture

Script processing

```
// Function to convert millimeters to inches
function convertMillimetersToInches(valueInMilimeters) {
    return valueInMilimeters/25.4;
}
```

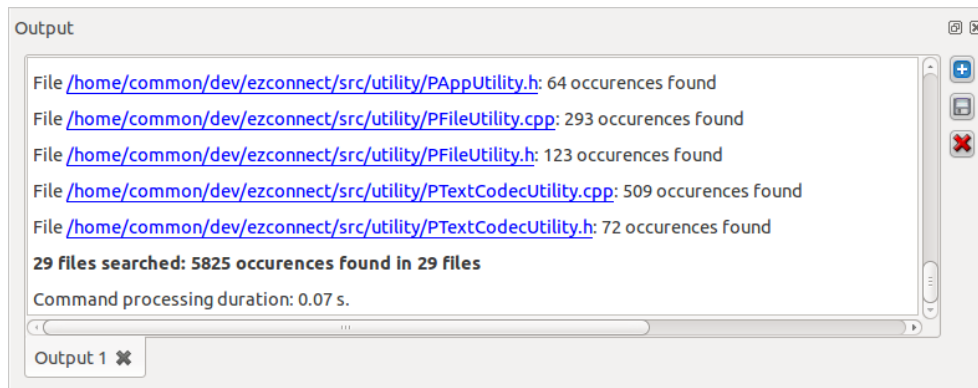
Finally copy and paste the following script within the script editor:

```
const NUMERICAL_VALUE_CAPTURE_INDEX = 1;
function convertUnits() {
    // We want to output a custom replacement for capture at index
    // NUMERICAL_VALUE_CAPTURE_INDEX (i. e. full numerical value)
    if(replaceCommand.captureIndex==NUMERICAL_VALUE_CAPTURE_INDEX) {
        // Convert the numerical value captured at current index
        // (i.e NUMERICAL_VALUE_CAPTURE_INDEX) from millimeters to inches
        var convertedValue =
            convertMillimetersToInches(replaceCommand.captureText);
        // Get the number of decimal digits to format the output
        // NB: Since version 2.0.6, the call to toString() on the returned
        // value by capturedTexts[] is not anymore required.
        var digitCount =
            replaceCommand.capturedTexts[NUMERICAL_VALUE_CAPTURE_INDEX+1].
            toString().length;
        // Take into account that the converted value is more than ten
        // times smaller, therefore we should at least add one digit to
        // represent it with a similar accuracy
        digitCount += 1;
        // Output the formatted value with an equivalent number of digits
        return convertedValue.toFixed(digitCount);
    }
    // For the other indexes (i.e. index!=NUMERICAL_VALUE_CAPTURE_INDEX)
    // we simply output the current replacement text
    return replaceCommand.replacementText;
}
// Function to convert millimeters to inches
function convertMillimetersToInches(valueInMilimeters) {
    return valueInMilimeters/25.4;
}
// Call to the utility function
convertUnits();
```

That's it! As mentioned above, we might remove the outer parentheses, then use `%0` as replacement pattern instead of `%1` and adjust the script accordingly.

6.11 Using the Output Window

The **Output** window is a multi-document preview for find and replace reports. Every time you search for an expression within files, a report will be outputted to the active console tab. The report includes links to files that contain searched/replaced expressions. A simple click on this link will open the file within [Find & Replace Preview](#). A double-click on this link will open the file with a suitable application.



The console showing a report for found expressions. The report gives you some statistics.

6.12 Editing the Preferences

The preferences window lets you customize some settings. This window is accessible via the menu **Edit/Preferences...** On Mac OS X, this entry is located in the application menu as usual.

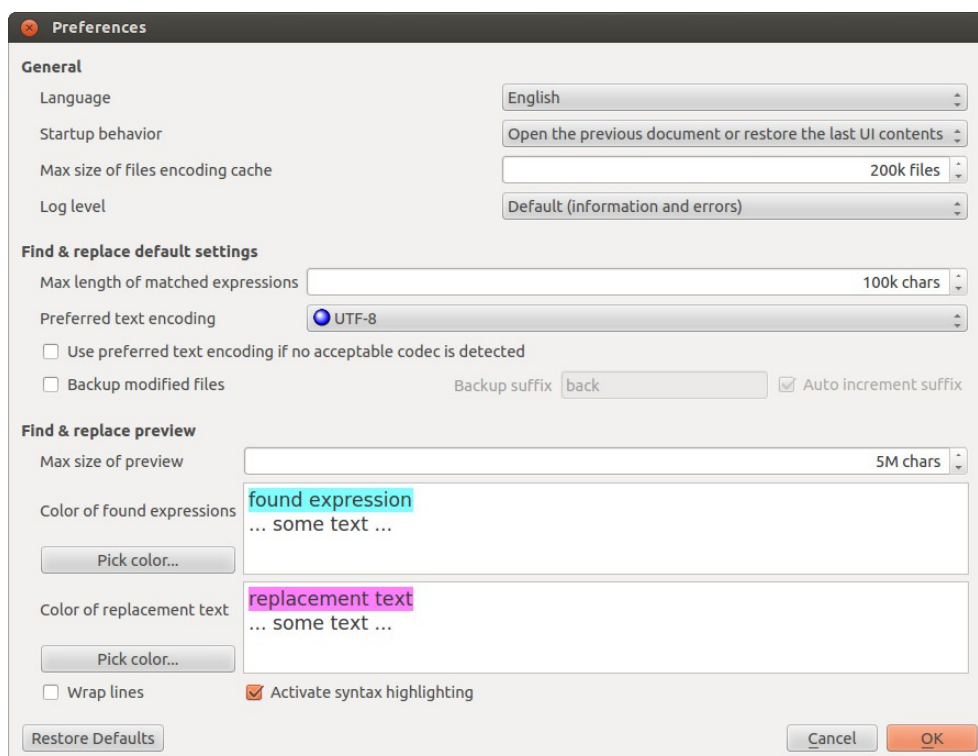
The available preferences are:

- **General**
 - **Language:** This field indicates the current language used for the user interface. You must restart the application for this change taking effect.
 - **Setup behavior:** This field indicates what is loaded by the software at startup time. There is the choice between the following options:
 - **Empty UI:** Nothing is loaded when starting the application. You start with a blank UI.
 - **Restore the last UI contents:** The application loads the previous contents for every controls in the UI.
 - **Open the last edited document:** The application opens the last document edited with the software, even if it was not during the last session (there isn't necessarily an edited document when using the application).
 - **Open the previous document or restore the last UI contents:** The application opens the document edited during the previous session if it exists; otherwise, it loads the previous contents for every controls in the UI.
 - **Max size of files encoding cache:** This field indicates the maximum number of files for which the detected encoding data are cached. The cache dramatically speeds up the encoding detection process. However, a very big cache can slow down the startup of the application, due to the time for loading cache. This time can usually be neglected in comparison to the performance increase during encoding detection.
 - **Log level:** This field tells the software what kinds of events must be logged in the log file. The log feature can be turned off by picking **No log**.
- **Find & replace default settings**

This section gives the defaults value used in the [Find & Replace Preferences](#) panel. Those default values are used every time the UI is reset: when closing the current [FRI document](#), when

creating a new document. Since those values are only default values for the **Find & Replace Preferences**, they do not directly affect the current behavior of the tools for searching and replacing text.

- **Max length of matched expressions:** See the chapter Editing Find & Replace Preferences [here](#), for more details.
- **Preferred text encoding:** See the chapter Changing the Encoding Detection Behavior [here](#), for more details.
- **Backup modified files, Backup suffix, Auto increment suffix:** See the chapter Doing Backups of Files before Modification [here](#), for more details.
- **Find & Replace preview**
 - **Max size of preview:** This field indicates the maximum number of chars that can be loaded in the preview window. If you try to load a file which is longer than this value, the end of the file will be ignored. For performance reasons, it is strongly recommended not to increase too much this value.
 - **Color of found expressions:** This editor shows the background color used to highlight found expressions.
 - **Color of replacement text:** This editor shows the background color used to highlight replaced text.
 - **Wrap lines:** Checking this box will cause words to be wrapped at the right edge of the text area. Wrapping occurs at whitespace, keeping whole words intact.
 - **Activate syntax highlighting:** Checking this box will activate syntax highlighting in the preview area. Syntax highlighting is only available for JavaScript, C/C++ and XML/HTML files.



The preferences window with default settings

In order to restore the default settings, simply click on **Restore Defaults**.

6.13 Using the Command Line

It is possible to execute replacement operations, encoding conversions as well as several administrative tasks directly from the command line. Here are the currently supported features.

Command	Action/Effect
-locale <locale_code>	Changes the locale used to <local_code>. E.g.: <i>FindReplaceIt -locale fr_CH</i> activates the Swiss French locale.
-reset-settings	Resets all settings: clears pre-loaded content, resets font settings, resets window positions, etc. The application is not launched.
-trace [<path/to/trace/file>.xml]	Enables tracing of application's activity. This enables advanced logging for debug purpose in case of problems. This might be useful for reporting a crash to our support team. Without the optional argument that specify a custom path for the trace file, this command will generate by default a trace file named <i>trace.xml</i> alongside the application executable file.
-version	Prints the full name and version of the application. The application is not launched.
<path/to/fri/file>.fri	Launches the application and opens the given file.
-replace <path/to/fri/file>.fri	Executes the replacement operation defined in the given file. This command produces a output that can be redirected to a file. The file has to be a complete FRI file , not the contents of a sub-window.
-batch-replace <path/to/fri/file>.fri	Executes the sequence of replacement operations defined in the given file. This command produces an output that can be redirected to a file. The file has to be a complete FRI file , not the contents of a sub-window.
-convert-encoding <path/to/fri/file>.fri	Executes the encoding conversion operation defined in the given file. This command produces an output that can be redirected to a file. The file has to be a complete FRI file , not the contents of a sub-window.
-find <path/to/fri/file>.fri	Executes the search operation defined in the given file. This command produces an output that can be redirected to a file. The file has to be a complete FRI file , not the contents of a sub-window.

In order to execute any of these commands from a terminal you only need to type the name of the application executable followed by the command. Examples:

```
./FindReplaceIt -reset-settings
```

For commands that produce a long output, you can redirect it into a file like this:

```
./FindReplaceIt -replace ~/documents/test/test-replace.fri > out.html
```

The command above writes into the file *out.html* the result produced by the replacement described in the file *test-replace.fri*.

When using the software from the command line, it might be very useful to use logical paths and




environment variables to locate the files to process. This is described in the chapter Using Logical Folders for Searching Files, [here](#).

Note: It is possible to use the application from the command line, however it requires a graphical environment.

6.14 Using Find & Replace It! Documents

Working with documents allows you to save and recover your work. However, it is possible to work with or without an active document within *Find & Replace It!*. By default there is no document created when the application is loaded. In other words there is no active document set when the application starts.

Here are the three ways of starting to work with an active document:



- Using the **File/New** menu  : starts a new document and resets the user interface
- Using the **File/Open...** menu  : the opened document becomes the current one
- Using the **File/Save** menu  : the current content of the user interface is saved, and the freshly saved document becomes the new current document


Having an active document has the following effects:


- Every time you try to quit the application, you'll be asked to save your modifications if any;
- Every time you open, or close a document, you'll be asked to save your modifications if any;
- The name of the current document is shown in the title bar of the main window;
- Using the **File/Save** menu will overwrite the document previously saved.

You might close the active document at any time by using the **File/Close** menu. This command will also reset the content of the user interface.

The documents generated by *Find & Replace It!* are suffixed by `.fri`. They are called **UI** files (i.e.: User Interface File) for *Find & Replace It!*.



All sections containing this button  support persistent serialization. Therefore, you can save an expression alongside with its replacement pattern and script, as well as all components for searching and filtering files. Of course the twin button  enables you to load a file previously saved.

As mentioned before, the **File** menu provides a way of saving and loading the full content of the interface. In fact, using the **File/Save** menu is almost equivalent to concatenate all files generated by all  buttons of all sub-windows.

If you open a `.fri` file from a sub-window  button (e.g.: [Find & Replace Editor](#)), only the content related to this sub-window will be loaded. Hence you can easily load only a part of a file saved from the **File/Save** menu. This also means that if you open a file generated from a sub-window inside another sub-window, nothing will be loaded. In the same way, opening a file generated from a specific sub-window with the **File/Save** menu, only loads the content of the specific sub-window, leaving all other windows intact.

7 Tips and Tricks

7.1 Multi-document Tabs

Some components use tabs to display multiple documents. To manually open a new document/tab, use this button . To close a document/tab, click on the button  located on the corresponding tab.

7.2 Working with Text Areas

7.2.1 Navigating text

The content of any text area can be navigated with the following key bindings:

Keypresses	Action
LeftArrow	Moves the cursor one character to the left.
Ctrl+LeftArrow	Moves the cursor one word to the left.
RightArrow	Moves the cursor one character to the right.
Ctrl+RightArrow	Moves the cursor one word to the right.
UpArrow	Moves the cursor one line up.
DownArrow	Moves the cursor one line down.
PageUp	Moves the cursor one page up.
PageDown	Moves the cursor one page down.
Home	Moves the cursor to the beginning of the line.
Ctrl+Home	Moves the cursor to the beginning of the text.
End	Moves the cursor to the end of the line.
Ctrl+End	Moves the cursor to the end of the text.
Ctrl+G	Moves the cursor to the beginning of a given line. This opens a popup for selecting the line number where one wants to jump to.

7.2.2 Editing Text

This is the list of key bindings which are implemented for editing:



Keypresses	Action
Backspace	Deletes the character to the left of the cursor.
Delete	Deletes the character to the right of the cursor.
Ctrl+C	Copy the selected text to the clipboard.
Ctrl+Insert	Copy the selected text to the clipboard.
Ctrl+K	Deletes to the end of the line.
Ctrl+V	Pastes the clipboard text into text edit.
Shift+Insert	Pastes the clipboard text into text edit.
Ctrl+X	Deletes the selected text and copies it to the clipboard.
Shift+Delete	Deletes the selected text and copies it to the clipboard.
Ctrl+Z	Undoes the last operation.

Keypresses	Action
Ctrl+Y	Redoes the last operation.
Alt+Wheel	Scrolls the page horizontally (the Wheel is the mouse wheel).
Ctrl++	Zooms in the text.
Ctrl--	Zooms in the text.
Tab (in some text areas)	Indents the current selection. Insert a tab if the selection is empty. This applies in the script editor and the previewed document editor only.
Shift+Tab (in some text areas)	Unindents the current selection. Insert a tab if the selection is empty. This applies in the script editor and the previewed document editor only.

To select (mark) text hold down the Shift key whilst pressing one of the movement keystrokes, for example, Shift+Right will select the character to the right, and Shift+Ctrl+Right will select the word to the right, etc.




7.2.3 Undo/Redo Changes

It is possible to undo and redo any change made in a text area when this area is editable. On the **Edit** tool bar or in the **Edit** menu, simply click on:

-  Undo (Ctrl+Z)
-  Redo (Ctrl+Y)

7.2.4 Changing Display Properties

The commands located in the **Text Display** menu let you change the appearance of the active text zone content:

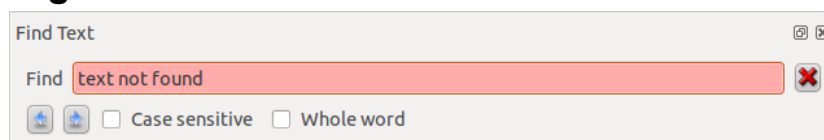
-  Zoom out (Ctrl++)
-  Zoom in (Ctrl+-)
-  Select a font


All these commands are available for all text area in the software but they only apply to the last area that has been activated. Therefore you might have to click somewhere inside a text area to get a result.

The default font and zoom can be restored with the **Text Display/Restore default font** command.

It is possible to print the content of the active text area through the **Text Display/Print...** command.

7.2.5 Searching for Text in Text Areas



It is possible to search for text within any text area of the graphical interface. Simply click on  in the **Edit** tool bar or in the **Edit** menu. This will show up the **Find Text** window. The search will occur in the last area that has been activated. The background is colorized in green when there is a match, in red when there is no match.

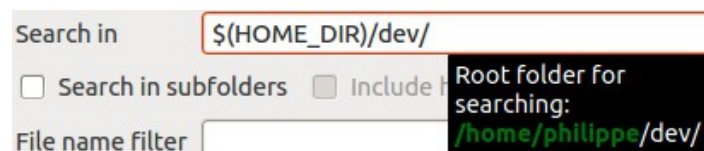
7.3 Using Logical Folders for Searching Files

Sometimes searching files in a folder given by its absolute path is not convenient. For instance it does not allow you to reuse your configuration ([saved in a .fri file](#)) on a different computer when files are not located in the same folder. Moreover, you might want to distribute .fri documents without showing your file system structure to people who receive the .fri files.

In order to specify a folder to search for files without actually having to type its absolute path we provide four logical folders accessible through the following variables:

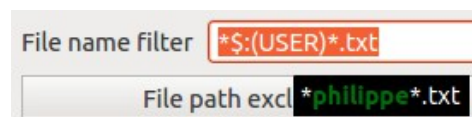
- **\$(APP_DIR)**: This variable refers to the directory's path that contains the application executable. It varies depending on where the software has been installed.
- **\$(CURRENT_DIR)**: This variable refers to the absolute path of the application's current directory.
- **\$(CURRENT_DOCUMENT_DIR)**: This variable refers to the absolute path of the current document's directory. If there is no .fri document opened, an empty path is returned.
- **\$(HOME_DIR)**: This variable refers to absolute path of the user's home directory.

Here is an example of how to use the variable **\$(HOME_DIR)**:



The tooltip shows the real file path obtained after evaluating the variable.

In a similar way it is possible to use environment variables within the path to search for files by using the placeholder **\$(var-name)**. The difference with the previous syntax is the presence of the **:** sign before the opening parenthesis. When a requested variable is not defined, the placeholder is replaced by an empty string. If the variable is not defined, the tooltip shows **\$(?var-name?)** (note the question marks around the name); otherwise, the tooltip shows the content with the value of each variable in bold and green font. It is also possible to use environment variables within the field **File name filter**. Note that the logical folders and their associated variables are not available in that typing field.



The tooltip shows the value of the environment variable named **USER**.

For instance, on Unix systems, it's possible to replace the **\$(HOME_DIR)** variable by the **HOME** variable, like this: **\$(HOME)/dev/ezconnect**. Of course this is less portable than the application variable **\$(HOME_DIR)**, since **HOME** is not defined on Windows systems.

Usage of environment variables is especially useful when using *Find & Replace It!* from the command line. That way you can automate string replacement jobs defined in any regular .fri files whatever the location of the files to process.

7.4 Using Local Storage for Settings

Instead of using the default location for the software's settings (in the user's home directory), it's possible to use a local (i.e. portable) folder for storing the user and the application settings.

Local settings are enabled by default as soon as it exists a folder named *settings* in the folder containing the executable of *Find & Replace It!*. Hence, you only need to create such a folder beside the executable file. See the installation instructions are available [here](#) for some additional details on the exact location.

This is useful for people who want to have a portable version of *Find & Replace It!* on USB key. Local settings might also be used in case one wants to share one install of the software, with the same user's preferences, on a network drive.

The *settings* folder contains one settings file with the user's preferences (shared on any machine where the software is executed) and one set of application files for each location where the software is run (activation key, log, encoding cache).

7.5 Multi File Selection in the Found Files List

To operate on many files at once it is possible to use the context menu on the **Found files** list. This menu let you act on the current selection. As shown below, possible actions on selected files are: toggling check marks, selecting encoding, loading files in the preview. Applying an action on a selection of files. Note that **Open file in test preview** will open all selected files as distinct documents within **Find & Replace Preview**.

	File path	Encoding		Created
8	<input checked="" type="checkbox"/> /home/common/dev/ezconne...ase/PSettingsManager.cpp	UTF-8	No	20.09.10 1
9	<input checked="" type="checkbox"/> /home/common/dev/ezconne...c/base/PSettingsManager.h	UTF-8	No	20.09.10 1
10	<input checked="" type="checkbox"/> /home/common/dev/ezconne...base/PrecompiledHeader.h	UTF-8	No	20.09.10 1
11	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/base/main.cpp	UTF-8	No	20.09.10 1
12	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/gui/PAboutDlg.cpp	UTF-8	No	20.09.10 1
13	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/gui/PAboutDlg.h	UTF-8	No	20.09.10 1
14	<input checked="" type="checkbox"/> /home/common/dev/ezconne...ui/PConnectionKeyDlg.cpp	UTF-8	No	20.09.10 1
15	<input checked="" type="checkbox"/> /home/common/dev/ezconne...c/gui/PConnectionKeyDlg.h	UTF-8	No	20.09.10 1
16	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/gui/PMainFrm.cpp	UTF-8	No	20.09.10 1
17	<input type="checkbox"/> /home/common/dev/ezconnect/src/gui/PMainFrm.h	UTF-8	No	20.09.10 1
18	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/twofish/AES.H	ISO-8859-1	No	20.09.10 1
19	<input type="checkbox"/> /home/common/dev/ezconnect/src/twofish/DEBUG.H	ISO-8859-1	None	20.09.10 1
20	<input checked="" type="checkbox"/> /home/common/dev/ezconnect/src/twofish/PLATFORM.H	ISO-8859-1	None	20.09.10 1

Open file in preview	Ctrl+Shift+P
Open file in default editor	Ctrl+Shift+D
Copy path to clipboard	
Select a codec for selection	Ctrl+Shift+T
Select preferred codec for selection	Ctrl+T
Check mark selection	Ctrl+Alt+C
Uncheck selection	Ctrl+Alt+U
Toggle check marks on selection	Space
Check mark exclusively the selection	Ctrl+K
Check mark all files	Ctrl+Shift+C
Uncheck all files	Ctrl+Shift+U
Refresh file list	F5
Detect file encoding	Ctrl+D


7.6 Getting Examples

There are some sample files shipped with *Find Replace It!*. Under Windows and Linux these files are located within the following directory:

`<application install dir>/res/examples`

On Mac OS X, these files are located in:

`<application install dir>/Find & Replace It!.app/Contents/Resources/examples`

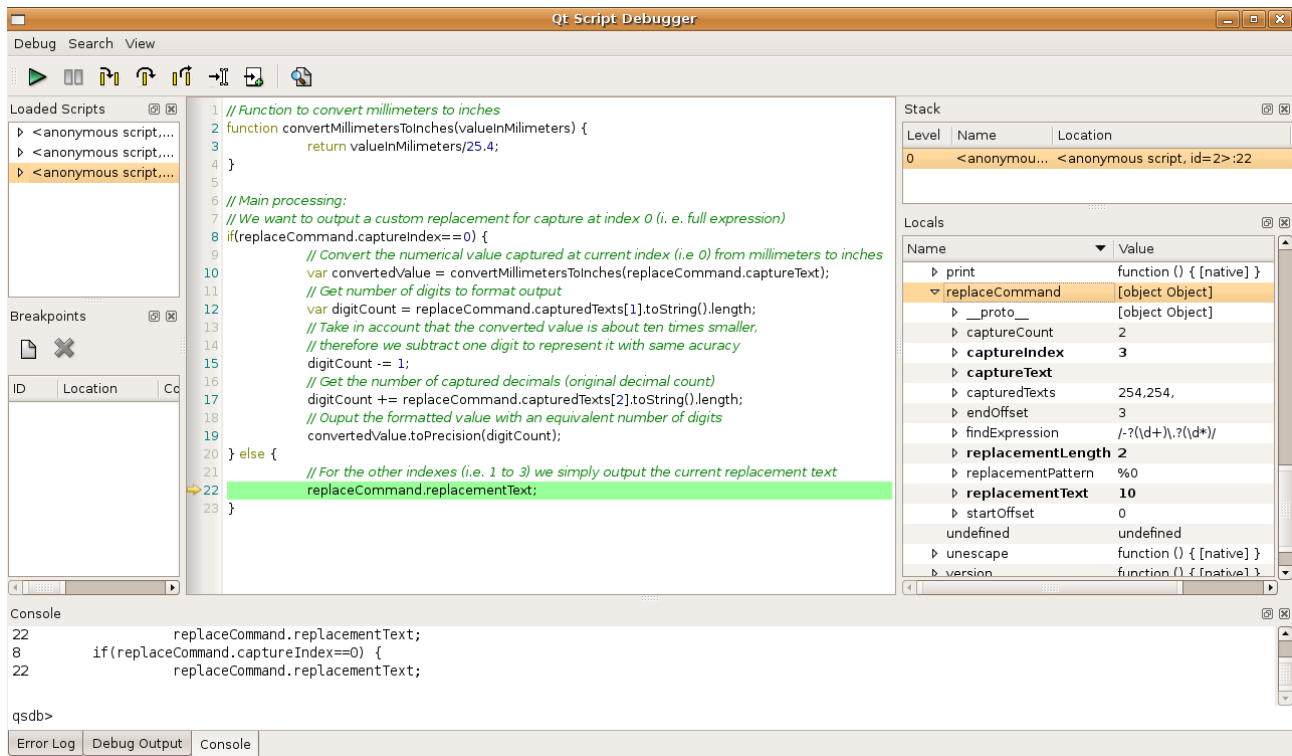
The files suffixed with *fri* are **UI** files (i.e.: [User Interface File](#)) for *Find & Replace It!*. Such files contain stored user interface data and can be opened with  buttons.

The files suffixed with *.txt* are sample data provided for convenience, in order to easily test the capabilities of *Find & Replace It!*.

Most examples provided alongside *Find & Replace It!* are available as a separate package which can be downloaded from <http://www.dprog.ch/home/download/> (see *Find & Replace It! Advanced Replacement Examples*).

7.7 Debugging Script

When willing to script some replacement texts it is convenient to debug the script. *Find & Replace It!* comes with an integrated debugger. To start the debugger, click on the **Execute in debugger** button located under the [script editor](#). The debugger will show up:



Script debugger in action. On the right we can see the 'Locals' dock window which displays the current context provided by the 'replaceCommand' object.

A user manual for script debugger is available at qt-project.org/doc/qt-4.8/qtscriptdebugger-manual.html

8 Regular Expressions¹

A regular expression, or "regex", is a pattern for matching substrings in a text. This is useful in many contexts, e.g.:

Searching	A regexp provides more powerful pattern matching than simple substring matching, e.g., match one of the words <i>mail</i> , <i>letter</i> or <i>correspondence</i> , but none of the words <i>email</i> , <i>mailman</i> , <i>mailer</i> , <i>letterbox</i> , etc.
Search and Replace	A regexp can replace all occurrences of a substring with a different substring, e.g., replace all occurrences of <i>&with &amp;</i> ; except where the <i>&</i> is already followed by an <i>amp</i> ;

The *Find & Replace It!* regexp is modeled on Perl's regexp language. It fully supports Unicode. The regexp can also be used in a simpler, **Wildcard** mode that is similar to the functionality found in command shells. The syntax rules used by regexp can be changed through the Syntax combo box. In particular, the pattern syntax can be set to **Simple text**, which means the pattern to be matched is interpreted as a plain string, i.e., special characters (e.g., backslash) are not escaped.

A good text on regexps is *Mastering Regular Expressions* (Third Edition) by Jeffrey E. F. Friedl, ISBN 0-596-52812-4.

8.1 Introduction

Regexps are built up from expressions, quantifiers, and assertions. The simplest expression is a character, e.g. *x* or *5*. An expression can also be a set of characters enclosed in square brackets. *[ABCD]* will match an *A* or a *B* or a *C* or a *D*. We can write this same expression as *[A-D]*, and an expression to match any capital letter in the English alphabet is written as *[A-Z]*.

A quantifier specifies the number of occurrences of an expression that must be matched. *x{1,1}* means match one and only one *x*. *x{1,5}* means match a sequence of *x* characters that contains at least one *x* but no more than five.

Note that in general regexps cannot be used to check for balanced brackets or tags. For example, a regexp can be written to match an opening html ** and its closing **, if the ** tags are not nested, but if the ** tags are nested, that same regexp will match an opening ** tag with the wrong closing **. For the fragment *bold bolder*, the first ** would be matched with the first **, which is not correct. However, it is possible to write a regexp that will match nested brackets or tags correctly, but only if the number of nesting levels is fixed and known. If the number of nesting levels is not fixed and known, it is impossible to write a regexp that will not fail.

Suppose we want a regexp to match integers in the range 0 to 99. At least one digit is required, so we start with the expression *[0-9]{1,1}*, which matches a single digit exactly once. This regexp matches integers in the range 0 to 9. To match integers up to 99, increase the maximum number of occurrences to 2, so the regexp becomes *[0-9]{1,2}*. This regexp satisfies the original requirement to match integers from 0 to 99, but it will also match integers that occur in the middle of strings. If we want the matched integer to be the whole string, we must use the anchor assertions, *^* (caret) and *\$* (dollar). When *^* is the first character in a regexp, it means the regexp must match from the beginning of the string. When *\$* is the last character of the regexp, it means the regexp must match to the end of the string. The regexp becomes *^[0-9]{1,2}\$*. Note that assertions, e.g. *^* and *\$*, do not match characters but locations in the string.

If you have seen regexps described elsewhere, they may have looked different from the ones shown here. This is because some sets of characters and some quantifiers are so common that they have been given special symbols to represent them. *[0-9]* can be replaced with the symbol

¹ This chapter was initially taken in 2009 from the Qt ® [documentation](#) from Nokia ®, available under LGPL. It as been adapted to fit the purpose of this manual.

`\d`. The quantifier to match exactly one occurrence, `{1,1}`, can be replaced with the expression itself, i.e. `x{1,1}` is the same as `x`. So our 0 to 99 matcher could be written as `^\d{1,2}$`. It can also be written `^\d\d{0,1}$`, i.e. *From the start of the string, match a digit, followed immediately by 0 or 1 digits*. In practice, it would be written as `^\d\d?$`. The `?` is shorthand for the quantifier `{0,1}`, i.e. 0 or 1 occurrences. `?` makes an expression optional. The regexp `^\d\d?$` means *From the beginning of the string, match one digit, followed immediately by 0 or 1 more digit, followed immediately by end of string*.

To write a regexp that matches one of the words *mail or letter or correspondence* but does not match words that contain these words, e.g., *email, mailman, mailer, and letterbox*, start with a regexp that matches *mail*. Expressed fully, the regexp is `m{1,1}a{1,1}i{1,1}l{1,1}`, but because a character expression is automatically quantified by `{1,1}`, we can simplify the regexp to `mail`, i.e., an *m* followed by an *a* followed by an *i* followed by an *l*. Now we can use the vertical bar `|`, which means “or”, to include the other two words, so our regexp for matching any of the three words becomes `mail|letter|correspondence`. Match *mail or letter or correspondence*. While this regexp will match one of the three words we want to match, it will also match words we don't want to match, e.g., *email*. To prevent the regexp from matching unwanted words, we must tell it to begin and end the match at word boundaries. First we enclose our regexp in parentheses, `(mail|letter|correspondence)`. Parentheses group expressions together, and they identify a part of the regexp that we wish to [capture](#). Enclosing the expression in parentheses allows us to use it as a component in more complex regexps. It also allows us to examine which of the three words was actually matched. To force the match to begin and end on word boundaries, we enclose the regexp in `\b word boundary` assertions: `\b(mail|letter|correspondence)\b`. Now the regexp means: *Match a word boundary, followed by the regexp in parentheses, followed by a word boundary*. The `\b` assertion matches a *position* in the regexp, not a *character*. A word boundary is any non-word character, e.g., a space, newline, or the beginning or ending of a string.

If we want to replace ampersand characters with the HTML entity `&`, the regexp to match is simply `&`. But this regexp will also match ampersands that have already been converted to HTML entities. We want to replace only ampersands that are not already followed by `amp;`. For this, we need the negative lookahead assertion, `(?!__)`. The regexp can then be written as `&(?!amp;)`, i.e. *Match an ampersand that is **not** followed by `amp;`*.

If we want to count all the occurrences of *Eric* and *Eirik* in a string, two valid solutions are `\b(Eric|Eirik)\b` and `\bEi?ri[ck]\b`. The word boundary assertion `\b` is required to avoid matching words that contain either name, e.g. *Ericsson*. Note that the second regexp matches more spellings than we want: *Eric, Erik, Eiric* and *Eirik*.

Some of the examples discussed above are implemented in the [examples](#) section.

Regexps can match case insensitively using the **Case sensitive** check box, and can use non-greedy matching when the **Minimal match** mark is checked.

8.2 Characters and Abbreviations for Sets of Characters

Element	Meaning
c	A character represents itself unless it has a special regexp meaning. e.g. c matches the character <i>c</i> .
\c	A character that follows a backslash matches the character itself, except as specified below. e.g., To match a literal caret at the beginning of a string, write <code>\^</code> .
\a	Matches the ASCII bell (BEL, 0x07).
\f	Matches the ASCII form feed (FF, 0x0C).
\n	Matches the ASCII line feed (LF, 0x0A, Unix newline).
\r	Matches the ASCII carriage return (CR, 0x0D).
\t	Matches the ASCII horizontal tab (HT, 0x09).

Element	Meaning
<code>\v</code>	Matches the ASCII vertical tab (VT, 0x0B).
<code>\xhhhh</code>	Matches the Unicode character corresponding to the hexadecimal number <i>hhhh</i> (between 0x0000 and 0xFFFF).
<code>\0ooo</code> (i.e., \zero ooo)	matches the ASCII/Latin1 character for the octal number <i>ooo</i> (between 0 and 0377).
<code>.</code> (dot)	Matches any character (including newline).
<code>\d</code>	Matches a digit.
<code>\D</code>	Matches a non-digit.
<code>\s</code>	Matches a whitespace character including line separators.
<code>\S</code>	Matches a non-whitespace character.
<code>\w</code>	Matches a word character (letters, numbers, marks and '_').
<code>\W</code>	Matches a non-word character.
<code>\n</code>	The <i>n</i> -th backreference , e.g. \1, \2, etc.

8.3 Sets of Characters

Square brackets mean match any character contained in the square brackets. The character set abbreviations described above can appear in a character set in square brackets. Except for the character set abbreviations and the following two exceptions, characters do not have special meanings in square brackets.

<code>^</code>	The caret negates the character set if it occurs as the first character (i.e. immediately after the opening square bracket). <code>[abc]</code> matches <i>a</i> or <i>b</i> or <i>c</i> , but <code>[^abc]</code> matches anything but <i>a</i> or <i>b</i> or <i>c</i> .
<code>-</code>	The dash indicates a range of characters. <code>[w-z]</code> matches <i>w</i> or <i>x</i> or <i>y</i> or <i>z</i> .

Using the predefined character set abbreviations is more portable than using character ranges across platforms and languages. For example, `[0-9]` matches a digit in Western alphabets but `\d` matches a digit in *any* alphabet.

Note: In other regexp documentation, sets of characters are often called "character classes".

8.4 Quantifiers

By default, an expression is automatically quantified by `{1,1}`, i.e. it should occur exactly once. In the following list, **E** stands for expression. An expression is a character, or an abbreviation for a set of characters, or a set of characters in square brackets, or an expression in parentheses.

<code>E?</code>	Matches zero or one occurrences of E . This quantifier means <i>The previous expression is optional</i> , because it will match whether or not the expression is found. <code>E?</code> is the same as <code>E{0,1}</code> . e.g., <code>dents?</code> matches <i>dent</i> or <i>dents</i> .
<code>E+</code>	Matches one or more occurrences of E . <code>E+</code> is the same as <code>E{1,}</code> . e.g., <code>0+</code> matches <i>0</i> , <i>00</i> , <i>000</i> , etc.
<code>E*</code>	Matches zero or more occurrences of E . It is the same as <code>E{0,}</code> . The <code>*</code> quantifier is often used in error where <code>+</code> should be used. For example, if <code>\s*\$</code> is used in an expression to match strings that end in whitespace, it will match every string because <code>\s*\$</code> means <i>Match zero or more white spaces followed by end of string</i> . The correct regexp to match strings that have at least one trailing whitespace character is <code>\s+\$</code> .
<code>E{n}</code>	Matches exactly <i>n</i> occurrences of E . <code>E{n}</code> is the same as repeating <i>E</i> <i>n</i> times. For example, <code>x{5}</code> is the same as <code>xxxxx</code> . It is also the same as <code>E{n,n}</code> , e.g. <code>x{5,5}</code> .
<code>E{n,}</code>	Matches at least <i>n</i> occurrences of E .
<code>E{,m}</code>	Matches at most <i>m</i> occurrences of E . <code>E{,m}</code> is the same as <code>E{0,m}</code> .

E{n,m}	Matches at least <i>n</i> and at most <i>m</i> occurrences of E .
---------------	--

To apply a quantifier to more than just the preceding character, use parentheses to group characters together in an expression. For example, `tag+` matches a `t` followed by an `a` followed by at least one `g`, whereas `(tag)+` matches at least one occurrence of `tag`.

Note: Quantifiers are normally "greedy". They always match as much text as they can. For example, `0+` matches the first zero it finds and all the consecutive zeros after the first zero. Applied to `20005`, it matches `20005`. Quantifiers can be made non-greedy through the check box **Minimal match**.

8.5 Capturing Text

Parentheses allow us to group elements together so that we can quantify and capture them. For example if we have the expression `mail|letter|correspondence` that matches a string we know that *one* of the words matched but not which one. Using parentheses allows us to "capture" whatever is matched within their bounds, so if we used `(mail|letter|correspondence)` and matched this regexp against the string `I sent you some email` we can use the `%x` replacement pattern to extract the matched characters, in this case `mail`.

We can use captured text within the regexp itself. To refer to the captured text we use "backreferences" which are indexed from 1, the same as for `%x`. For example we could search for duplicate words in a string using `\b(w+)\W+!1\b` which means match a word boundary followed by one or more word characters followed by one or more non-word characters followed by the same text as the first parenthesized expression followed by a word boundary.

If we want to use parentheses purely for grouping and not for capturing we can use the non-capturing syntax, e.g. `(?:green|blue)`. Non-capturing parentheses begin `(?:` and end `)`. In this example we match either `green` or `blue` but we do not capture the match so we only know whether or not we matched but not which color we actually found. Using non-capturing parentheses is more efficient than using capturing parentheses since the regexp engine has to do less book-keeping.

Captured text can be accessed in replacement pattern using `%0` which returns the full expression match, or using `%i` (with $1 < i \leq 9$) which returns the captured string at the given index.

Both capturing and non-capturing parentheses may be nested.

8.6 Assertions

Assertions make some statement about the text at the point where they occur in the regexp but they do not match any characters. In the following list **E** stands for any expression.

^	The caret signifies the beginning of the string. If you wish to match a literal <code>^</code> you must escape it by writing <code>\^</code> . For example, <code>^#include</code> will only match strings which <i>begin</i> with the characters <code>#include</code> . (When the caret is the first character of a character set it has a special meaning, see Sets of Characters .)
\$	The dollar signifies the end of the string. For example <code>\d\s*\$</code> will match strings which end with a digit optionally followed by whitespace. If you wish to match a literal <code>\$</code> you must escape it by writing <code>\\$</code> .
\b	A word boundary. For example the regexp <code>\bOK\b</code> means match immediately after a word boundary (e.g. start of string or whitespace) the letter <code>O</code> then the letter <code>K</code> immediately before another word boundary (e.g. end of string or whitespace). But note that the assertion does not actually match any whitespace so if we write <code>(\bOK\b)</code> and we have a match it will only contain <code>OK</code> even if the string is <code>It's OK now</code> .
\B	A non-word boundary. This assertion is true wherever <code>\b</code> is false. For example if we searched for <code>\Bon\b</code> in "Left on" the match would fail (space and end of string aren't

	non-word boundaries), but it would match in <i>tonne</i> .
(?=E)	Positive lookahead. This assertion is true if the expression matches at this point in the regexp. For example, <code>const(=?\s+char)</code> matches <i>const</i> whenever it is followed by <i>char</i> , as in <i>static const char *</i> . (Compare with <code>const\s+char</code> , which matches <i>static const char *</i>).
(?!E)	Negative lookahead. This assertion is true if the expression does not match at this point in the regexp. For example, <code>const(?!\s+char)</code> matches <i>const</i> except when it is followed by <i>char</i> .

8.7 Wildcard Matching

Most command shells such as *bash* or *cmd.exe* support "file globbing", the ability to identify a group of files by using wildcards. The Syntax combo box is used to switch between regexp and wildcard mode. Wildcard matching is much simpler than full regexps and has only four features:

c	Any character represents itself apart from those mentioned below. Thus <i>c</i> matches the character <i>c</i> .
?	Matches any single character. It is the same as <i>.</i> in full regexps.
*	Matches zero or more of any characters. It is the same as <i>.*</i> in full regexps.
[...]	Sets of characters can be represented in square brackets, similar to full regexps. Within the character class, like outside, backslash has no special meaning.

In the mode Wildcard, the wildcard characters cannot be escaped. In the mode Wildcard Unix, the character *"* escapes the wildcard.

For example if we are in wildcard mode and have strings which contain filenames we could identify HTML files with **.html*. This will match zero or more characters followed by a dot followed by *h*, *t*, *m* and *l*.

Wildcard matching can be convenient because of its simplicity, but any wildcard regexp can be defined using full regexps, e.g. `.*\s+html?$.` Notice that we can't match both *.html* and *.htm* files with a wildcard unless we use `*.htm*` which will also match *test.html.bak*. A full regexp gives us the precision we need, `.*\s+html?$.`

8.8 Notes for Perl Users

Most of the character class abbreviations supported by Perl are supported by regexp's, see [characters and abbreviations for sets of characters](#).

In regexps, apart from within character classes, *^* always signifies the start of the string, so carets must always be escaped unless used for that purpose. In Perl the meaning of caret varies automagically depending on where it occurs so escaping it is rarely necessary. The same applies to *\$* which in regexps always signifies the end of the string.

Regexp's quantifiers are the same as Perl's greedy quantifiers. Non-greedy matching cannot be applied to individual quantifiers, but can be applied to all the quantifiers in the pattern. For example, to match the Perl regexp `ro+?m` requires: `ro+m` and **Minimal match**=`true`

The equivalent of Perl's `/i` option is **Case sensitive** check box turned on.

In regexp *.* matches any character, therefore all regexps have the equivalent of Perl's `/s` option. Regexp does not have an equivalent to Perl's `/m` option, but this can be emulated in various ways for example by splitting the input into lines or by looping with a regexp that searches for newlines.

Because regexp is string oriented, there are no `\A`, `\Z`, or `\z` assertions. The `\G` assertion is not supported but can be emulated in a loop.

Perl's `$&` is `%0`. There are no regexp equivalents for `$``, `$'` or `$+`. Perl's capturing variables, `$1`, `$2`, ... correspond to `\1`, `\2` inside search pattern and `%1`, `%2` inside replacement pattern, etc.

Perl's extended `/x` syntax is not supported, nor are directives, e.g. `(?i)`, or regexp comments, e.g. `(?#comment)`.

Both zero-width positive and zero-width negative look-ahead assertions `(?=pattern)` and `(?!pattern)` are supported with the same syntax as Perl. Perl's look-behind assertions, "independent" sub-expressions and conditional expressions are not supported.

Non-capturing parentheses are also supported, with the same `(?:pattern)` syntax.

8.9 Examples

<code>^\d\d?\$</code>	
<i>Match integers from 0 to 99</i>	
<code>123</code>	Do not match
<code>-6</code>	Do not match
<code>6</code>	Match

The third string matches `6`. This is a simple validation regexp for integers in the range 0 to 99.

<code>^\s+\$</code>	
<i>Match strings without white spaces</i>	
<code>Hello world</code>	Do not match
<code>This_is-OK</code>	Match

The second string matches `This_is-OK`. We've used the character set abbreviation `\S` (non-whitespace) and the anchors to match strings which contain no whitespace.

In the following example we match strings containing `mail` or `letter` or `correspondence` but only match whole words i.e. not `email`.

<code>\b(mail letter correspondence)\b</code>	
<i>Match words mail, letter and correspondence</i>	
<code>I sent you an email</code>	Do not match
<code>Please write the letter</code>	Match

The second string matches `Please write the letter`. The word `letter` is also captured (because of the parentheses). We can see what text we've captured like this: `%1 = \1 = letter`

This will capture the text from the first set of capturing parentheses (counting capturing left parentheses from left to right). The parentheses are counted from 1 since `%0 (\0)` is the whole matched regexp (equivalent to `&` in most regexp engines).

<code>& (?!amp;)</code>	
<i>Match ampersands but not &amp; ;</i>	
<code>This & that</code>	Match one occurrence at index 6
<code>His & ; hers & theirs</code>	Match one occurrence at index 16

9 Licensing Information

9.1 End User Licenses

The terms “This Software” below refer to *Find & Replace It!*.

9.1.1 End User License for NON App Store Customers

By installing and/or using this Software, you are confirming your acceptance of the Software and agreeing to become bound by the terms of the agreement as specified in the EULA:

http://www.dprog.ch/multimedia/findreplaceit-docs/2.1/license_en.html.

Find & Replace It! is protected by an activation system. Hence to get a license for the full version of the product you should activate your product with a serial code called **Activation Key**. Without the activation, the software can still be run in demo mode with some limitations. You'll find more information in the [Activation chapter](#).

The license will last forever, provided you do not reinstall your copy of the software on a different hardware configuration (see *Important Notes* below).

The license covers all minor updates (e.g.: all 0.x.y releases). Some major updates (e.g.: from 0.x to 1.x) might be included in the license but without guarantee.

Important Notes:

- One license allows you to install the software on only one computer at one time.
- A license is valid for only one type of operating system (i.e.: Linux, Mac or Windows).
- We kindly request people who have two computers running the same type of operating system (either Linux, Mac or Windows) to contact our support to get a second key for free.
- If your hardware configuration has changed, you might require an update of your license in order to reactivate the product. Such an update is totally free of charge.
- All demands regarding the licenses updates should be addressed to: support[at]dprog[dot]ch.

9.1.2 End User License for App Store Customers

If you buy this Software on the App Store from Apple, you must confirm your acceptance of the Software and agree to become bound by the terms of the agreement as specified in the [EULA provided by Apple](#).

For people who bought *Find & Replace It!* on the App Store, there is no activation key required to use the software, but there is no demo version available either.

9.2 Third Party Licenses and Credits

This section contains information on the different licenses *Find & Replace It!* is shipped with, and credits to some of our many contributors.

This Software uses the [Qt open source libraries](http://qt-project.org/doc/qt-4.8/lgpl.html) from Digia Plc. This library is distributed under the GNU Lesser General Public License version 2.1 plus some exceptions (see qt-project.org/doc/qt-4.8/lgpl.html). The Qt binaries and source code are available for download at <http://qt-project.org/downloads>. You are free to use your own versions of the Qt dynamic libraries instead of the ones shipped with the application. See the *Customizing Installation* section of <http://www.dprog.ch/multimedia/findreplaceit-docs/2.1/install.html#title-customizing-installation> for more details. You are permitted to modify this Software for your own use only. Reverse engineering for debugging such modifications is allowed. For more details about licenses used by Qt please read the following links:

- qt-project.org/doc/qt-4.8/licenses.html
- qt-project.org/doc/qt-4.8/3rdparty.html

This Software uses the [ICU open source library](http://source.icu-project.org/repos/icu/icu/trunk/license.html) from IBM Corporation. ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software. ICU is released under a nonrestrictive open source license that is suitable for use with both commercial software and with other open source or free software (see <http://source.icu-project.org/repos/icu/icu/trunk/license.html>).

This Software uses the Syntax Highlighters plugins from Philippe Docourt. This is a collection of Qt plugins for managing syntax highlighting. It allows you to dynamically add syntax highlighting capabilities to your Qt applications. Currently it supports syntax highlighting for qscript, javascript, C++, Java, C#, and html/xml documents. This collection of plugins is distributed under the GNU Lesser General Public License version 2.1 (see <http://www.gnu.org/licenses/lgpl-2.1.txt>). The source code of these plugins are available for download at <http://www.dprog.net/products-topmenu-29/syntax-highlighters>.

The Linux installer uses an auto-extractable archive generated by makeself. makeself.sh is a small shell script developed by Stéphane Peter that generates a self-extractable archive (see <http://megastep.org/makeself>).

The Windows installer is generated by NSIS (Nullsoft Scriptable Install System), an open source system to create Windows installers (see http://nsis.sourceforge.net/Main_Page).

This Software is built with the Qt Integration Scripts package from Philippe Docourt (see <http://www.dprog.net/products-topmenu-29/qt-integration-scripts>).

Most of the icons used in this Software are free icons, provided under either the GNU Lesser General Public License v2.1 (see <http://www.gnu.org/licenses/lgpl-2.1.txt>) or in the public domain. A special thanks to the authors:

- The Tango project members, authors of Tango icon theme: http://tango.freedesktop.org/Tango_Desktop_Project
- Everaldo Coelho, author of Christal icon collection: <http://www.everaldo.com/crystal/>
- David Vignoni, author of Nuvola icon set: <http://www.icon-king.com/projects/nuvola/>
- Digia Plc, author of Qt icons: <http://qt.digia.com/>

9.3 Trademarks

- dProg, the dProg logo, *Find & Replace it!* and the *Find & Replace It!* logo are trademarks of dProg - Philippe Docourt in Switzerland.
- Digia, the Digia logo, Qt, and the Qt logo are trademarks of Digia Plc and/or its subsidiaries in Finland and other countries.
- Intel, Intel Inside (logos), MMX and Pentium are ® trademarks of Intel Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Linux is a ® trademark of Linus Torvalds in the United States, other countries or both.
- Mac, Mac OS, App Store and Macintosh are ® trademarks of Apple Computer, Inc., registered in the U.S. and other countries.
- Microsoft, Windows, Windows NT, XP, Visual Studio, Word and the Windows logo are ® trademarks of Microsoft Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- All other company, product, or service names may be trademarks or service marks of others and are the property of their respective owners.